به نام دادار دادآفرین

مینی پروژه سوم درس مبانی سیستم‌های هوشمند

استاد درس: دکتر مهدی علیاری

گردآورنده: امیر جهانگرد تکالو

شماره دانشجویی: ۹۹۲۵۲۹۳

۱۳۰۷

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق

فهرست

سوال ۱

با استفاده از کران مرتبه اول (رابطهٔ ۴-۱۱ در [۲]) و کران مرتبه دوم (رابطهٔ ۱۱-۱۰ در [۲])، دو سیستم فازی با غیرفازی‌ساز میانگین و ماکزیمم طراحی کنید که تابع $g\left(x_1, x_2\right) = \frac{1}{3+x_1+x_2}$ روی $U = [-1,1] \times [-1,1]$ را به‌شکل یکنواخت و با دقت $\epsilon = 0.1$ تقریب بزند. سیستم‌های فازی طراحی‌شده را رسم کرده و با هم مقایسه کنید.

بخش اول: کران مرتبه اول با غیرفازی ساز میانگین

در قسمت اول قصد داریم سیستم فازی $f(x)$ را با استفاده از کران مرتبهٔ اول طراحی کنیم. فرض می‌کنیم که تابع $g(x_1, x_2)$ روی مجموعهٔ $U = [-1,1] \times [-1,1] \subset \mathbb{R}^2$ تعریف شود. ما می‌خواهیم سیستم فازی $f(x)$ را طوری طراحی کنیم که تابع $g(x)$ را با دقت ϵ تقریب بزند. بیان ریاضیاتی این موضوع به شرح رابطهٔ ۴ است.

$$\|g - f\|_\infty = \sup_{x \in U} |g(x) - f(x)| \leq \left\| \frac{\partial g}{\partial x_1} \right\|_\infty h_1 + \left\| \frac{\partial g}{\partial x_2} \right\|_\infty h_2 \leq \epsilon \tag{1}$$

حال برای طراحی سیستم فازی به‌صورت گام‌به‌گام پیش می‌رویم:

گام اول: در این مثال، تعداد N_i $(i = 1,2)$ مجموعهٔ فازی در بازهٔ $U = [\alpha_i, \beta_i] = [-1,1] \subset \mathbb{R}^2$ به‌صورت $A^1, \ldots, A_i^N$ و با توابع تعلق مثلثیِ $\mu_{A_i^1}\left(x_i; a_i^1, b_i^1, c_i^1\right), \ldots, \mu_{A_i}^{N_i}\left(x_i; a_i^{N_i}, b_i^{N_i}, c_i^{N_i}\right)$ تعریف می‌کنیم. هم‌چنین در نظر داریم که: $A_i^1 < A_i^2 < \cdots < A_i^{N_i}$، $b_i^1 = \alpha_i = -1$، $b_i^{N_i} = \beta_i = 1$. از سوی دیگر، اگر مرکز مجموعهٔ فازی A_i^1 را با e_i^1 نشان دهیم، خواهیم داشت: $e_i^1 = \alpha_i = -1$. هم‌چنین اگر مرکز مجموعهٔ فازی $A_i^{N_i}$ را با $e_i^{N_i}$ نشان دهیم خواهیم داشت: $e_i^{N_i} = \beta_i = 1$. بقیهٔ مراکز را هم به‌صورت $e_i^j = \alpha_i + h_i(j-1)$ در نظر می‌گیریم. هم‌چنین از آن‌جا که نقطهٔ شروع تابع بعدی، مرکز تابع قبلی است، فرض می‌کنیم که: $a_i^{k+1} = b_i^k$.

گام دوم: در ادامه، $M = N_1 \times N_2$ قاعدهٔ اگر-آنگاه فازی را به‌صورت «اگر x_1، $A_1^{i_1}$ و x_2، $A_2^{i_2}$ باشد؛ آن‌گاه y، $B^{i_1 i_2}$ است.» می‌سازیم. لازم به ذکر است که $i_q = 1, 2, \ldots, N_q (q = 1, 2)$ بوده و مراکز مجموعه‌های فازی $B^{i_1 i_2}$ را با $\bar{y}^{i_1 i_2} = g\left(e_1^{i_1}, e_2^{i_2}\right)$ نشان می‌دهیم. این مراکز با توجه به روابط اصلی مطرح‌شده در صورت مسأله به‌صورت زیر تعریف می‌شوند:

$$y^{-i_1 i_2} = g\left(e_1^{i_1}, e_2^{i_2}\right) = \frac{3}{e_1^{i_1}+e_2^{i_2}} \tag{۲}$$

گام سوم: سیستم فازی $f(x)$ را از قواعد گام دوم و با بهره‌گیری از موتور استنتاج ضرب، فازی‌ساز منفرد و غیرفازی‌ساز میانگین مراکز تشکیل می‌دهیم:

$$f(x) = \frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2} \bar{y}^{i_1 i_2}\left[\mu_{A_1^{i_1}}(x_1)\,\mu_{A_2^{i_2}}(x_2)\right]}{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\left[\mu_{A_1^{i_1}}(x_1)\,\mu_{A_2^{i_2}}(x_2)\right]} = \frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2} g\left(e_1^{i_1}, e_2^{i_2}\right)\left[\mu_{A_1}(x_1)\,\mu_{A_2}^{i_2}(x_2)\right]}{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\left[\mu_{A_1^{i_1}}(x_1)\,\mu_{A_2^{i_2}}(x_2)\right]} \tag{۳}$$

محاسبه دقت تقریب سیستم‌های فازی:

$$\|g-f\|_\infty = \sup_{x\in U}|g(x)-f(x)| \le \left\|\frac{\partial g}{\partial x_1}\right\|_\infty h_1 + \left\|\frac{\partial g}{\partial x_2}\right\|_\infty h_2 \le \epsilon, \begin{cases} \left\|\frac{\partial g}{\partial x_1}\right\|_\infty = \sup_{x\in U}\left|\frac{\partial g}{\partial x_i}\right| \\ h_i = \max_{1\le j\le N_{i-1}}\left|e_i^{j+1} - e_i^j\right| \end{cases} \tag{۴}$$

با توجه به فرض دقت $\epsilon = ۰/۱$ برای تقریب سیستم فازی و هم‌چنین فرض $h_1 = h_2 = h$ داریم:

$$\varepsilon > h\left(\left\|\frac{\partial g}{\partial x_1}\right\|_\infty + \left\|\frac{\partial g}{\partial x_2}\right\|_\infty\right) \to h < \frac{\varepsilon}{\left\|\frac{\partial g}{\partial x_1}\right\|_\infty + \left\|\frac{\partial g}{\partial x_2}\right\|_\infty} \tag{۵}$$

پس حال می‌توان h را حساب کرد:

$$\left\|\frac{\partial g}{\partial x_1}\right\| = \left\|\frac{\partial g}{\partial x_2}\right\| = \sup_{x\in U}\left|\frac{\partial g}{\partial x_1}\right| = \sup_{x\in U}\left|\frac{\partial g}{\partial x_2}\right| = \sup_{x\in U}\left|\frac{-1}{(3+x_1+x_2)^2}\right|$$

$$\to x_1 = x_2 = -1 \to \left\|\frac{\partial g}{\partial x_1}\right\| = \left\|\frac{\partial g}{\partial x_2}\right\| = 1$$

$$\to 1\times h + 1\times h \le 0.1 \to h \le 0.05 \to h = 0.05$$

حال با داشتن h می‌توانیم تعداد توابع تعلق را بدست بیاوریم:

$$n = \frac{b-a}{h} = \frac{1-(-1)}{0.05} = \frac{2}{0.05} = 40$$

$$\to N_1 = N_2 = n+1 = 41$$

پس در مجموع ۴۱ مجموعه فازی با توابع تعلق مثلثی بصورت زیر خواهیم داشت:

$$\mu_{A^1}(x) = \mu_{A^1}(x; a_1, b_1, c_1) = \mu_{A^1}(x; -1, -1, -1 + h)$$

$$\mu_{A^j}(x) = \mu_{A^j}\left(x; a_j, b_j, c_j\right) = \mu_{A^j}\left(x; e^{j-1}, e^j, e^{j+1}\right), \begin{cases} j = 2, \dots, 40 \\ e^j = a + h(j-1) = -1 + 0.05(j-1) \end{cases}$$

$$\mu_{A^{41}}(x) = \mu_{A^{41}}(x; a_{41}, b_{41}, c_{41}) = \mu_{A^{41}}(x; 1 - h, 1, 1)$$

پس در مجموع $N_1 \times N_2 = 41 \times 41 = 1681$ قاعده اگر-آنگاه فازی خواهیم داشت و در نهایت سیستم فازی f(x) بصورت زیر محاسبه می‌شود:

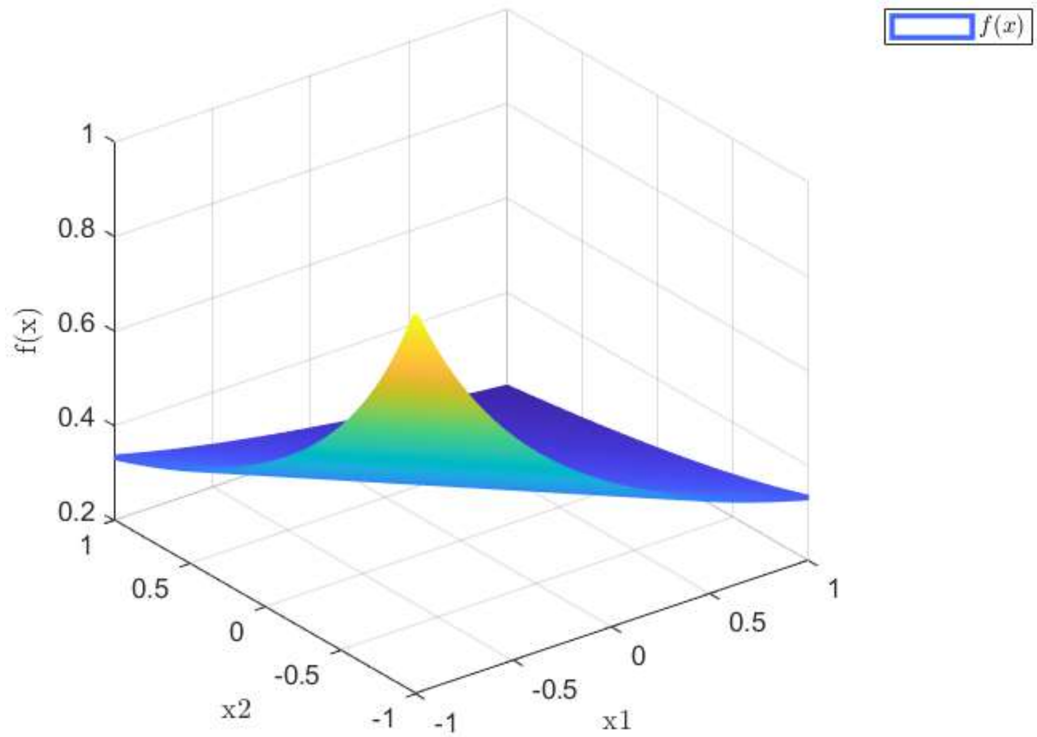
$$f(x) = \frac{\sum_{i1=1}^{41} \sum_{i2=1}^{41} g\left(e_1^{i_1}, e_2^{i_2}\right) \left[\mu_{A_1^{i1}}(x_1) \mu_{A_2^{i2}}(x_2)\right]}{\sum_{i1=1}^{41} \sum_{i2=1}^{41} \left[\mu_{A_1^{i1}}(x_1) \mu_{A_2^{i2}}(x_2)\right]}$$

کد متلب مربوط به محاسبات بالا((f(x و g(مراکز مجموعه‌های فازی)):

```
clc;
clear all;
close all;

tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.05;
N = 41;

g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

[x1,x2] = meshgrid(x1,x2);

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
        e_i1(i1,1) = -1 + h*(i1-1);
        e_i2(i2,1) = -1 + h*(i2-1);
```

```
        if i1==1
            mu_A_x1 = trimf(x1, [-1,-1,-1+h]);
        elseif i1==N
            mu_A_x1 = trimf(x1,[1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1,[-1+h*(i1-2), -1+h*(i1-1), -
1+h*(i1)]);
        end

        if i2==1
            mu_A_x2 = trimf(x2, [-1,-1,-1+h]);
        elseif i2==N
            mu_A_x2 = trimf(x2,[1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2,[-1+h*(i2-2), -1+h*(i2-1), -
1+h*(i2)]);
        end


       g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
       num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
       den=den+mu_A_x1.*mu_A_x2;
       k=k+1;
   end
end

f_x = num./den;
g_x = 1./(3+x1+x2);
```

تابع f(x) تقریب‌زده شده در فضای سه بعدی بصورت زیر است:

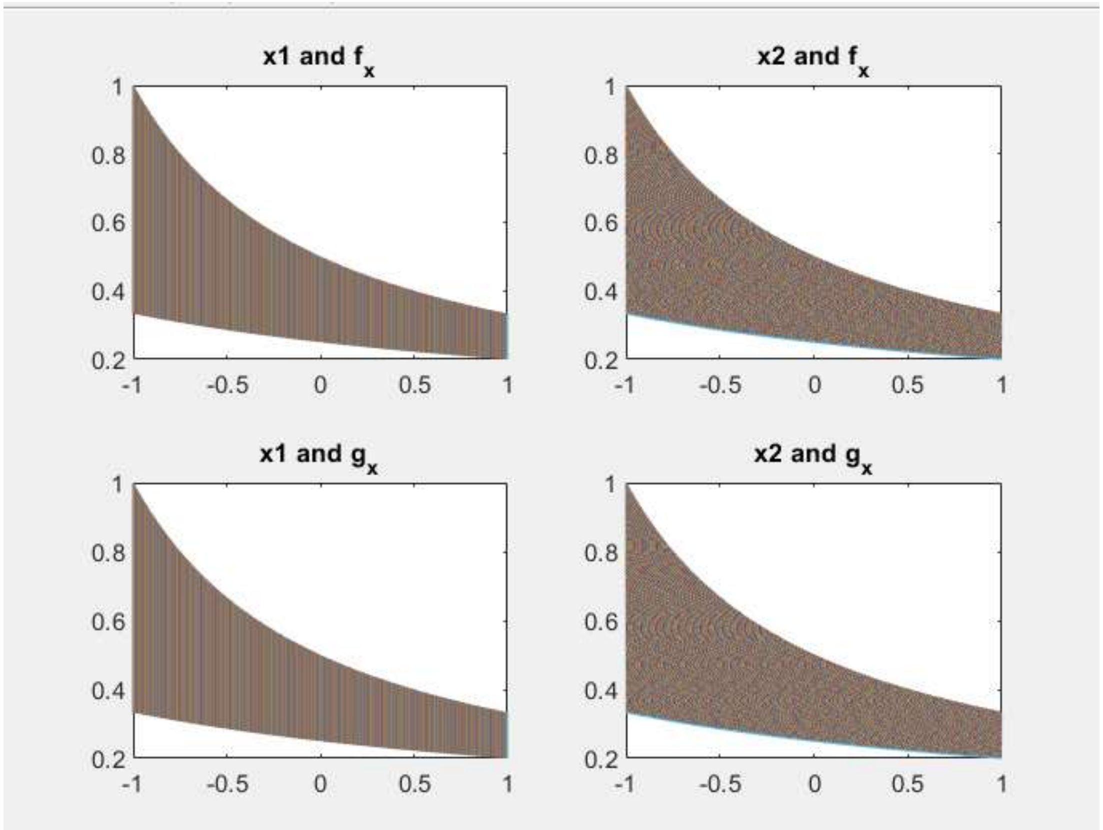


دستور متلب مربوط به این قسمت:

```
figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on
```

بررسی خروجی: این نمودار تابع تقریب زده شده را بر حسب x1 و x2 نشان می‌دهد. این نمودار به طور بصری نشان می‌دهد که تابع یک قله دارد، که نشان می‌دهد ممکن است در برش دوبعدی یا ممکن است یک تابع گوسی در زمینه دو متغیر x1 و x2 باشد. تابع با افزایش فاصله از قله به هر دو سمت در صفحه ایجاد شده توسط x1 و x2 کاهش می‌یابد.

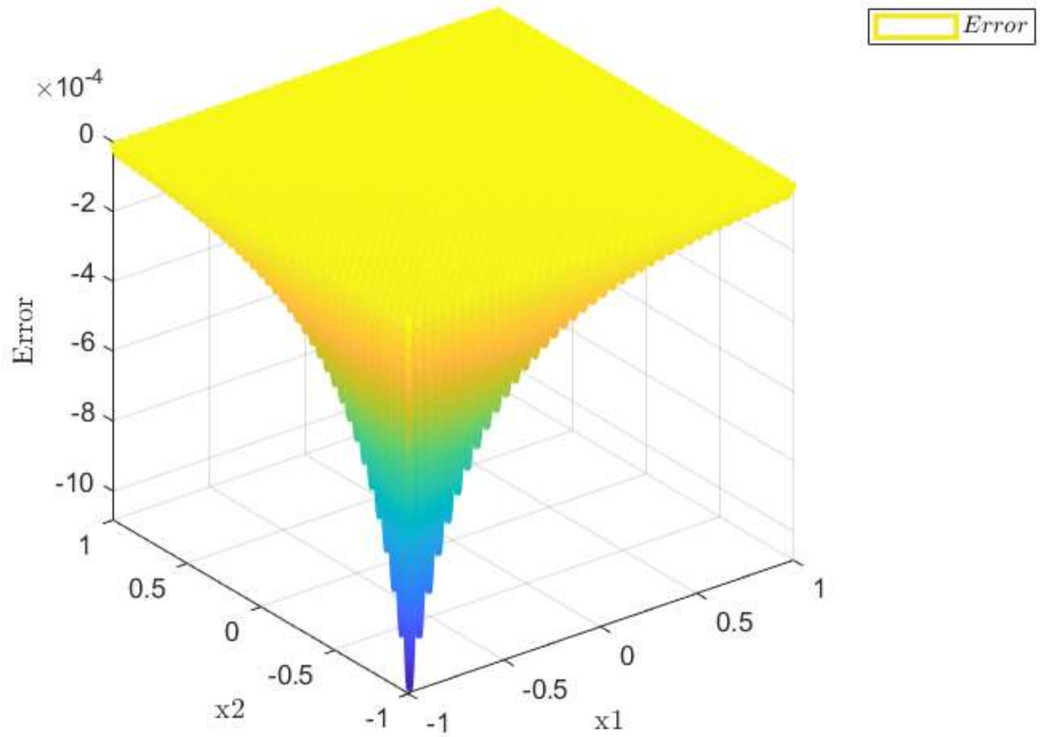
رسم سیستم اصلی g(x) و سیستم تقریب‌زده‌شده f(x) در فضای ۲ بعدی بر حسب x1 و x2 :



دستور متلب مربوط به این قسمت:

```
figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");

subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");
```

همچنین میزان خطا بین سیستم اصلی و سیستم تقریب‌زده شده بدین صورت می‌باشد:



کد متلب مربوطه:

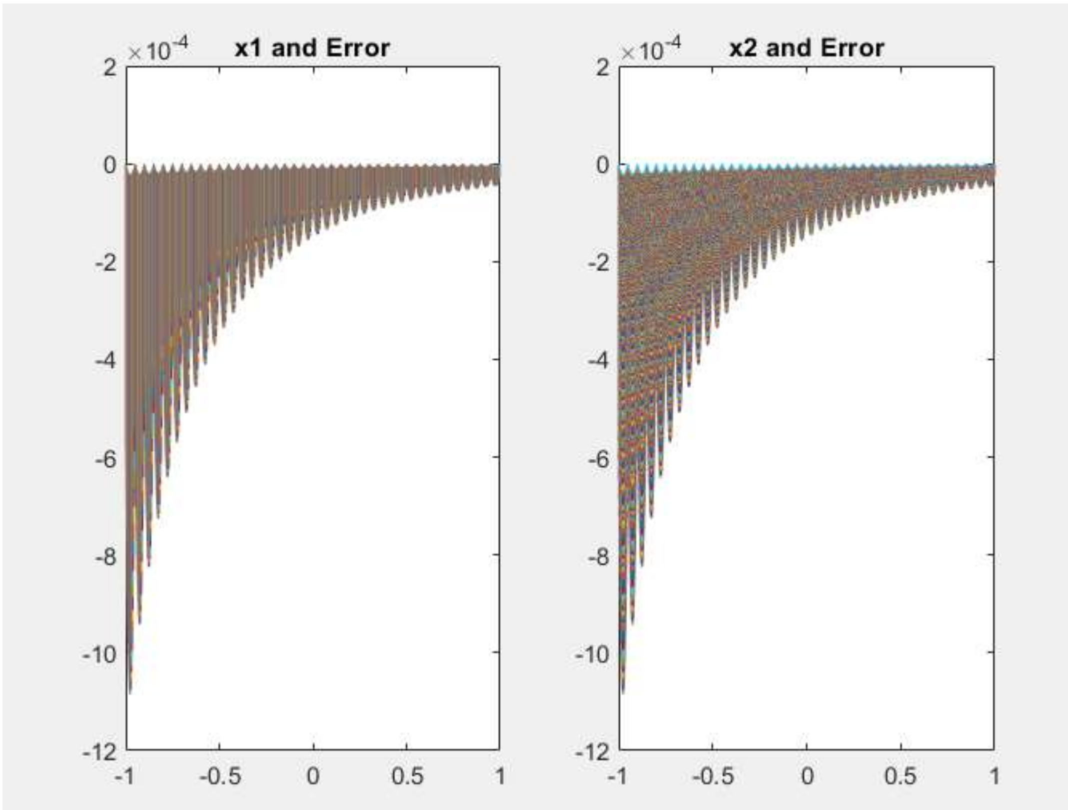
```
figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on
```

تحلیل شکل: سطح نمودار یک شکل مخروطی با قله‌ای به سمت پایین است. این نشان می‌دهد که Error مقدار حداقلی در مرکز یا نزدیک مرکز دارد که x1 و x2 در آنجا به صفر می‌رسند. خطا هر چه دورتر از مرکز در هر جهت در صفحه x1-x2 حرکت کنیم، افزایش می‌یابد.

تحلیل رنگ‌ها: نمودار از یک گرادیان از رنگ‌ها استفاده می‌کند که از آبی در پایین مخروط (حداقل خطا) به قرمز و سپس زرد هستند که نشان می‌دهد خطا افزایش می‌یابد. این طرح رنگ‌ها معمولاً برای تأکید بصری بر تغییرات مقادیر استفاده می‌شود، با رنگ‌های "سرد" که معمولاً مقادیر پایین را نشان می‌دهند و رنگ‌های "گرم" که مقادیر

بالاتر را نشان می‌دهند. طبق جعبه زرد در بالای شکل و سمت راست که با برچسب Error وجود دارد، متوجه می-شویم که رنگ سطح مقدار خطا که زرد است بالاترین خطا را دارد.

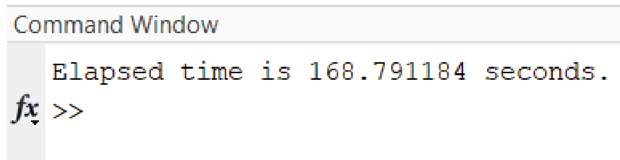
میزان خطا در فضای ۲ بعدی:



کد متلب مربوط به نمودار:

```
figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");

subplot(1,2,2);
plot(x2,E);
title("x2 and Error");
```

محاسبه مدت زمانی که طول می‌کشد تا برنامه خروجی دهد:



شبیه‌سازی در متلب(همه دستورات متلب مربوط به این قسمت یک جا در اینجا جمع شده است):

```
clc;
clear all;
close all;

tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.05;
N = 41;

g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

[x1,x2] = meshgrid(x1,x2);

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
        e_i1(i1,1) = -1 + h*(i1-1);
        e_i2(i2,1) = -1 + h*(i2-1);
        if i1==1
            mu_A_x1 = trimf(x1, [-1,-1,-1+h]);
        elseif i1==N
            mu_A_x1 = trimf(x1,[1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1,[-1+h*(i1-2), -1+h*(i1-1), -
1+h*(i1)]);
        end

        if i2==1
```

```
            mu_A_x2 = trimf(x2, [-1,-1,-1+h]);
        elseif i2==N
            mu_A_x2 = trimf(x2,[1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2,[-1+h*(i2-2), -1+h*(i2-1), -
1+h*(i2)]);
        end


       g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
       num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
       den=den+mu_A_x1.*mu_A_x2;
       k=k+1;
   end
end

f_x = num./den;
g_x = 1./(3+x1+x2);


figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on

figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");

subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");
```

```
figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");

subplot(1,2,2);
plot(x2,E);
title("x2 and Error");

toc
```

بخش دوم: کران مرتبه دوم با غیرفازی ساز میانگین

در ادامه ضمن حفظ گام‌های طراحی مشترک، دقت تقریب سیستم فازی را بوسیلهٔ قضیهٔ مربوط به **کران مرتبه دوم** تعیین می‌کنیم. این قضیه این‌گونه بیان می‌دارد که اگر فرض کنیم $f(x)$ یک سیستم فازی مطابق رابطهٔ ۳ باشد و $g(x)$ روی بازهٔ $U = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2]$ تا دو مرتبه به‌صورت پیوسته مشتق‌پذیر باشد؛ آن‌گاه داریم:

$$\|g(x)-f(x)\|_\infty \leq \frac{1}{8}\left[\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty h_1^2 + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty h_2^2\right] \leq \epsilon, \qquad \begin{cases} \left\|\frac{\partial^2 g}{\partial x_i^2}\right\|_\infty = \sup_{x\in U}\left|\frac{\partial^2 g}{\partial x_i^2}\right| \\ h_i = \max_{1\leq j\leq N_i-1}\left|e_i^{j+1} - e_i^j\right| (i = 1, 2) \end{cases}$$

با دقت $\varepsilon = 0.1$ و فرض $h_1 = h_2 = h$ داریم:

$$h^2 < \frac{8\varepsilon}{\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty} \rightarrow h < \sqrt{\frac{8\varepsilon}{\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty}}$$

$$\left\|\frac{\partial^2 g}{\partial {x_1}^2}\right\| = \left\|\frac{\partial^2 g}{\partial {x_2}^2}\right\| = \sup_{x\in U}\left|\frac{\partial^2 g}{\partial {x_1}^2}\right| = \sup_{x\in U}\left|\frac{\partial^2 g}{\partial {x_2}^2}\right| = \sup_{x\in U}\left|\frac{2}{(3+x_1+x_2)^3}\right|$$

$$\rightarrow x_1 = x_2 = -1 \rightarrow \left\|\frac{\partial^2 g}{\partial {x_1}^2}\right\| = \left\|\frac{\partial^2 g}{\partial {x_2}^2}\right\| = 2$$

$$\rightarrow \frac{1}{8}(2\times h + 2\times h) \le 0.1 \rightarrow h \le 0.2 \rightarrow h = 0.2$$

حال با داشتن h می‌توانیم تعداد توابع تعلق را بدست بیاوریم:

$$n = \frac{b-a}{h} = \frac{1-(-1)}{0.2} = \frac{2}{0.2} = 10$$

$$\rightarrow N_1 = N_2 = n+1 = 11$$

پس در مجموع ۱۱ مجموعه فازی با توابع تعلق مثلثی بصورت زیر خواهیم داشت:

$$\mu_{A^1}(x) = \mu_{A^1}(x; a_1, b_1, c_1) = \mu_{A^1}(x; -1, -1, -1+h)$$

$$\mu_{A^j}(x) = \mu_{A^j}\left(x; a_j, b_j, c_j\right) = \mu_{A^j}\left(x; e^{j-1}, e^j, e^{j+1}\right), \begin{cases} j = 2, \ldots, 10 \\ e^j = a + h(j-1) = -1 + 0.2(j-1) \end{cases}$$

$$\mu_{A^{11}}(x) = \mu_{A^{11}}(x; a_{11}, b_{11}, c_{11}) = \mu_{A^{11}}(x; 1-h, 1, 1)$$

پس در مجموع $N_1 \times N_2 = 11 \times 11 = 121$ قاعده اگر-آنگاه فازی خواهیم داشت و در نهایت سیستم فازی f(x) بصورت زیر محاسبه می‌شود:

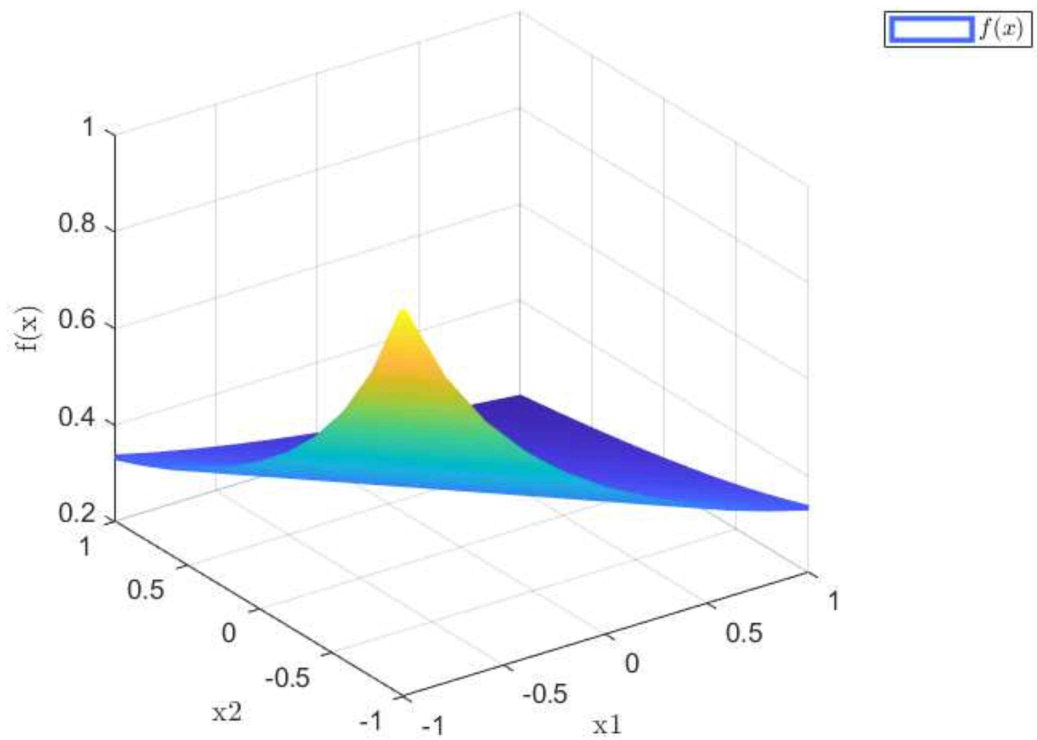
$$f(x) = \frac{\sum_{i1=1}^{11}\sum_{i2=1}^{11} g\left(e_1^{i_1}, e_2^{i_2}\right)\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}{\sum_{i1=1}^{11}\sum_{i2=1}^{11}\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}$$

کد متلب مربوط به محاسبات بالا((f(x) و g(مراکز مجموعه‌های فازی)):

```
%% center average defuzzifier with second order bound
clc
clear all
close all
tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.2;
N = 11;
```

```
g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

[x1,x2] = meshgrid(x1,x2);

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
    e_i1(i1,1) = -1 + h*(i1-1);
    e_i2(i2,1) = -1 + h*(i2-1);
        if i1==1
            mu_A_x1 = trimf(x1, [-1,-1,-1+h]);
        elseif i1==N
            mu_A_x1 = trimf(x1,[1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1,[-1+h*(i1-2), -1+h*(i1-1), -
1+h*(i1)]);
        end

        if i2==1
            mu_A_x2 = trimf(x2, [-1,-1,-1+h]);
        elseif i2==N
            mu_A_x2 = trimf(x2,[1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2,[-1+h*(i2-2), -1+h*(i2-1), -
1+h*(i2)]);
        end


       g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
       num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
       den=den+mu_A_x1.*mu_A_x2;
       k=k+1;
       end
end

f_x = num./den;
g_x = 1./(3+x1+x2);
```

تابع f(x) تقریب‌زده شده در فضای سه بعدی بصورت زیر است:

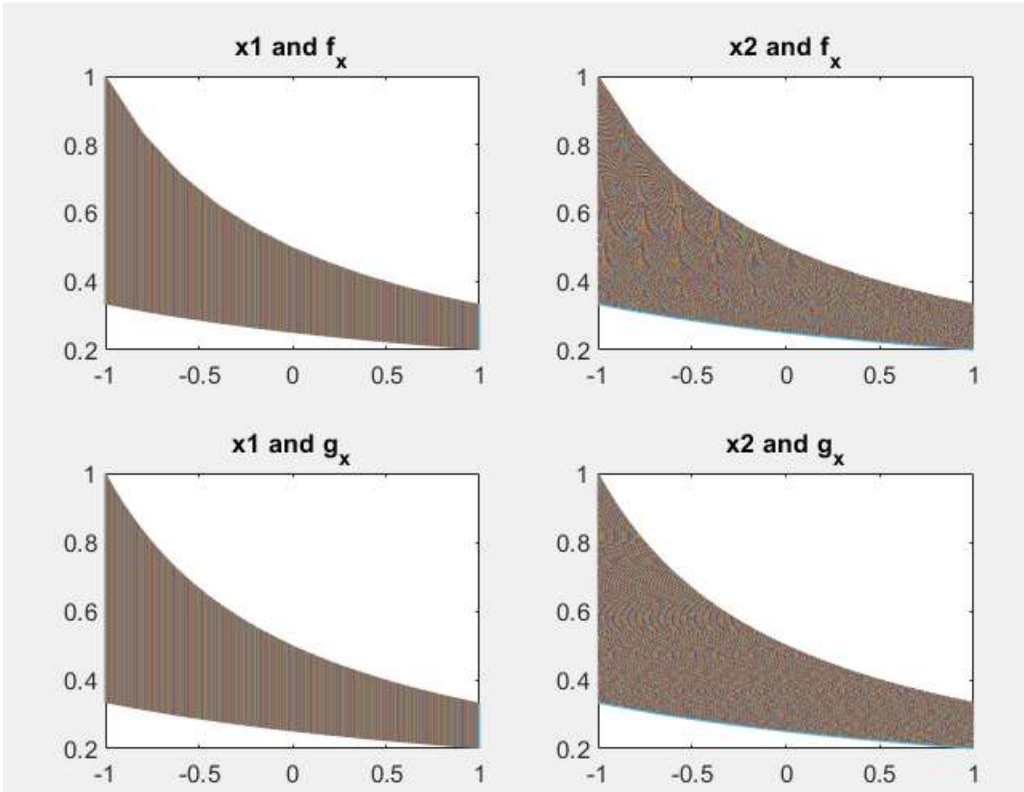


دستور متلب مربوط به این قسمت:

```
figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on
```

بررسی خروجی: این نمودار تابع تقریب زده شده را بر حسب x1 و x2 نشان می‌دهد. این نمودار به طور بصری نشان می‌دهد که تابع یک قله دارد، که نشان می‌دهد ممکن است در برش دوبعدی یا ممکن است یک تابع گوسی در زمینه دو متغیر x1 و x2 باشد. تابع با افزایش فاصله از قله به هر دو سمت در صفحه ایجاد شده توسط x1 و x2 کاهش می‌یابد.

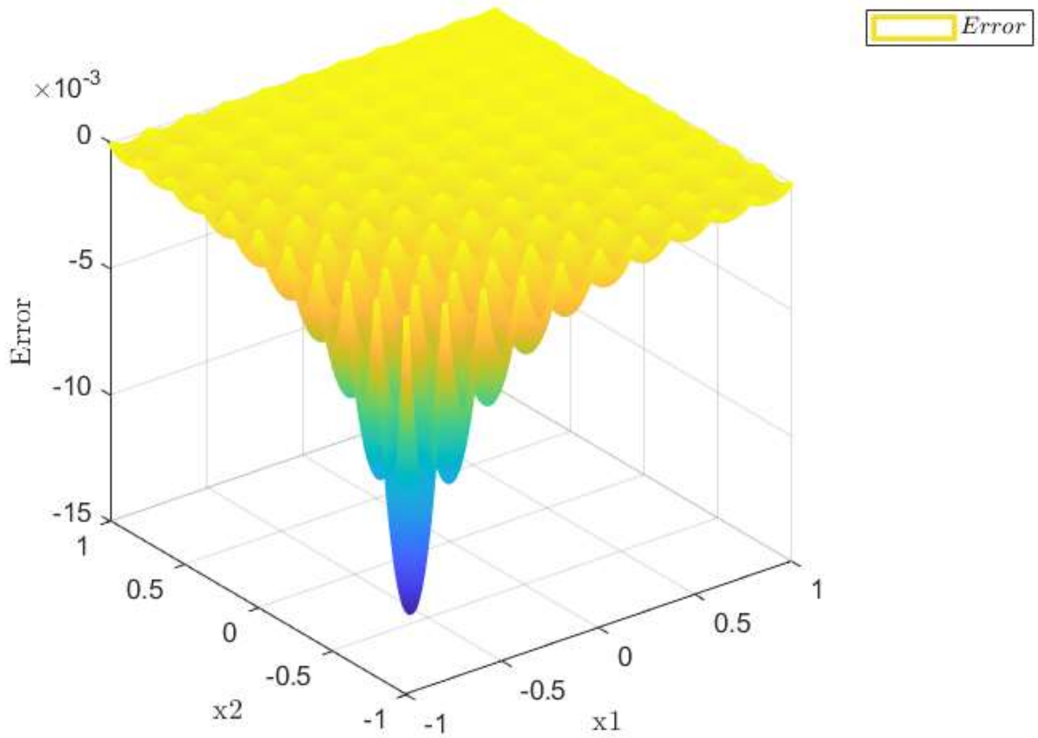
رسم سیستم اصلی g(x) و سیستم تقریب‌زده‌شده f(x) در فضای ۲ بعدی بر حسب x1 و x2 :



دستور متلب مربوط به این قسمت:

```
figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");

subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");
```

همچنین میزان خطا بین سیستم اصلی و سیستم تقریب‌زده شده بدین صورت می‌باشد:



کد متلب مربوطه:

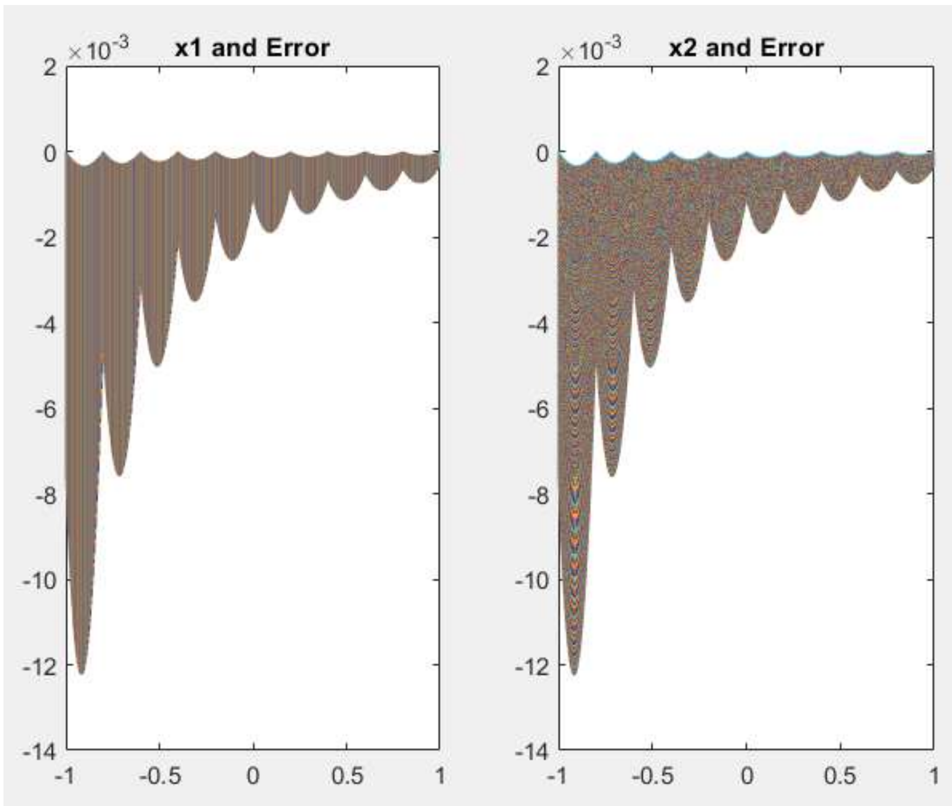
```
figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on
```

تحلیل شکل: سطح نمودار یک شکل مخروطی با قله‌ای به سمت پایین است. این نشان می‌دهد که Error مقدار حداقلی در مرکز یا نزدیک مرکز دارد که x1 و x2 در آنجا به صفر می‌رسند. خطا هر چه دورتر از مرکز در هر جهت در صفحه x1-x2 حرکت کنیم، افزایش می‌یابد.

تحلیل رنگ‌ها: نمودار از یک گرادیان از رنگ‌ها استفاده می‌کند که از آبی در پایین مخروط (حداقل خطا) به قرمز و سپس زرد هستند که نشان می‌دهد خطا افزایش می‌یابد. این طرح رنگ‌ها معمولاً برای تأکید بصری بر تغییرات

مقادیر استفاده می‌شود، با رنگ‌های "سرد" که معمولاً مقادیر پایین را نشان می‌دهند و رنگ‌های "گرم" که مقادیر بالاتر را نشان می‌دهند. طبق جعبه زرد در بالای شکل و سمت راست که با برچسب Error وجود دارد، متوجه می‌شویم که رنگ سطح مقدار خطا که زرد است بالاترین خطا را دارد.

میزان خطا در فضای ۲ بعدی:



کد متلب مربوط به نمودار:

```
figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");

subplot(1,2,2);
plot(x2,E);
title("x2 and Error");
```

محاسبه مدت زمانی که طول می‌کشد تا برنامه خروجی دهد:

```
Command Window
Elapsed time is 15.070416 seconds.
fx >>
```

مقایسه کران مرتبه اول و کران مرتبه دوم:

هر دو تقریب، خروجی‌های مناسبی برای تقریب ما هستند ولی چون اولا سرعت کران مرتبه دوم بالاتر است و زودتر خروجی را به ما می‌دهد و ثانیا چون در کران مرتبه دوم از توابع تعلق کمتری استفاده کردیم، پس تقریب با کران مرتبه دوم برای ما مناسب‌تر می‌باشد.

شبیه‌سازی در متلب(همه دستورات متلب مربوط به این قسمت یک جا در اینجا جمع شده است):

```
%% center average defuzzifier with second order bound
clc
clear all
close all
tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.2;
N = 11;

g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

[x1,x2] = meshgrid(x1,x2);

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
    e_i1(i1,1) = -1 + h*(i1-1);
    e_i2(i2,1) = -1 + h*(i2-1);
        if i1==1
            mu_A_x1 = trimf(x1, [-1,-1,-1+h]);
        elseif i1==N
```

```
            mu_A_x1 = trimf(x1,[1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1,[-1+h*(i1-2), -1+h*(i1-1), -
1+h*(i1)]);
        end

        if i2==1
            mu_A_x2 = trimf(x2, [-1,-1,-1+h]);
        elseif i2==N
            mu_A_x2 = trimf(x2,[1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2,[-1+h*(i2-2), -1+h*(i2-1), -
1+h*(i2)]);
        end


       g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
       num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
       den=den+mu_A_x1.*mu_A_x2;
       k=k+1;
       end
end

f_x = num./den;
g_x = 1./(3+x1+x2);


figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on

figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");
```

```
subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");

figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");

subplot(1,2,2);
plot(x2,E);
title("x2 and Error");

toc
```

بخش سوم: کران مرتبه اول با غیرفازی ساز ماکسیمم

گام اول و دوم طراحی مشابه قسمت‌های قبل می‌باشد که از تکرار آن صرف نظر می‌کنیم و سراغ گام سوم می-رویم(مشابه حل مرجع):

گام سوم: سیستم فازی $f(x)$ را از $N_1 \times N_2$ قاعده (۲-۱۱) و با استفاده از موتور استنتاج ضرب، فازی‌ساز منفرد و غیرفازی‌ساز ماکزیمم (مطابق لم ۴-۹ مرجع [۱۱]) تشکیل دهید.

$$f(x) = \bar{y}^{i_1^* i_2^*} = g\left(e_1^{i_1^*}, e_2^{i_2^*}\right) = \frac{1}{3 + e_1^{i_1^*} + e_2^{i_2^*}} \tag{۱۱-۶۲}$$

که در آن $i_1^* i_2^*$ به گونه‌ای است که

$$\mu_{A_1^{i_1^*}}(x_1)\mu_{A_2^{i_2^*}}(x_2) \geq \mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2) \tag{۱۱-۶۳}$$

که $i_1 = 1,2,\ldots,N_1$ و $i_2 = 1,2,\ldots,N_2$ می‌باشد. حال دقت تقریب سیستم فازی را به وسیله قضیه زیر تعیین می‌کنیم (کران مرتبه اول):

قضیه: فرض کنید $f(x)$ سیستم فازی مطابق رابطه (۶۲-۱۱) و $g(x)$ بر روی $U = [\alpha_1,\beta_1] \times [\alpha_2,\beta_2]$ بطور پیوسته مشتق‌پذیر باشد؛ آنگاه

$$\|g(x) - f(x)\|_\infty \leq \left\|\frac{\partial g}{\partial x_1}\right\|_\infty h_1 + \left\|\frac{\partial g}{\partial x_2}\right\|_\infty h_2 \tag{۱۱-۶۴}$$

که در آن نرم بی‌نهایت $\left\|\frac{\partial g}{\partial x_i}\right\|_\infty$ بدین ترتیب تعریف می‌گردد: $\left\|\frac{\partial g}{\partial x_i}\right\|_\infty = sup_{x \in U}\left|\frac{\partial g}{\partial x_i}\right|$ و $h_i = max_{1 \leq j \leq N_{i-1}}\left|e_i^{j+1} - e_i^j\right|$ $(i = 1,2)$. با توجه به تمرین ۱-۱۱، $\left\|\frac{\partial g}{\partial x_1}\right\|_\infty = \left\|\frac{\partial g}{\partial x_2}\right\|_\infty = 1$ بدست آمده است و هم چنین پارامتر $h_1 = h_2 = h = 0.04$ انتخاب گردید. در نتیجه مشابه تمرین ۱۱-۱، ۵۱ مجموعه فازی A^m $(m = 1,2,\ldots,51)$ بر روی $[-1,1]$ با توابع تعلق مثلثی روابط (۱۴-۱۱) تا (۱۶-۱۱) تعریف می‌کنیم که در آن $j = 2,\ldots,50$ و $e^j = \alpha + h[j-1] = -1 + 0.04(j-1)$ می‌باشد. در ادامه سیستم فازی $f(x)$ را با $N_1 \times N_2 = 51 \times 51 = 2601$ قاعده اگر - آنگاه فازی به صورت رابطه (۲-۱۱) می‌سازیم.

فرض کنید که $U^{i_1 i_2} = \left[e_1^{i_1}, e_1^{i_1+1}\right] \times \left[e_2^{i_2}, e_2^{i_2+1}\right]$ که در آن $i_1 = 1,2,\ldots,N_1 - 1$، $i_2 = 1,2,\ldots,N_2 - 1$. از آنجا که

$$[\alpha_i,\beta_i] = [e_i^1, e_i^2] \cup [e_i^2, e_i^3] \cup \ldots \cup \left[e_i^{N_n-1}, e_i^{N_n}\right] \tag{۱۱-۶۵}$$

بنابراین U را بصورت زیر تقسیم‌بندی می‌کنیم:

$$U = [\alpha_1,\beta_1] \times [\alpha_2,\beta_2] = \cup_{i_1=1}^{N_1-1} \cup_{i_2=1}^{N_2-1} U^{i_1 i_2} \tag{۱۱-۶۶}$$

پس برای هر $x \in U$ یک $U^{i_1 i_2}$ وجود دارد به نحوی که $x \in U^{i_1 i_2}$ باشد. حال فرض کنید که $x \in U^{i_1 i_2}$ به این معنی که

$$x_1 \in \left[e_1^{i_1}, e_1^{i_1+1}\right], x_2 \in \left[e_2^{i_2}, e_2^{i_2+1}\right] \tag{۱۱-۶۷}$$

حال $U^{i_1 i_2}$ را به چهار زیر بخش کوچکتر بدین شکل تقسیم‌بندی می‌کنیم:

$$U^{i_1 i_2} = U_{00}^{i_1 i_2} \cup U_{01}^{i_1 i_2} \cup U_{10}^{i_1 i_2} \cup U_{11}^{i_1 i_2} \qquad (۱۱\text{-}۶۸)$$

که در آن

$$U_{00}^{i_1 i_2} = \left[e_1^{i_1}, \frac{1}{2}\left(e_1^{i_1} + e_1^{i_1+1}\right)\right] \times \left[e_2^{i_2}, \frac{1}{2}\left(e_2^{i_2} + e_2^{i_2+1}\right)\right] \qquad (۱۱\text{-}۶۹)$$

$$U_{01}^{i_1 i_2} = \left[e_1^{i_1}, \frac{1}{2}\left(e_1^{i_1} + e_1^{i_1+1}\right)\right] \times \left[\frac{1}{2}\left(e_2^{i_2} + e_2^{i_2+1}\right), e_2^{i_2+1}\right] \qquad (۱۱\text{-}۷۰)$$

$$U_{10}^{i_1 i_2} = \left[\frac{1}{2}\left(e_1^{i_1} + e_1^{i_1+1}\right), e_1^{i_1+1}\right] \times \left[e_2^{i_2}, \frac{1}{2}\left(e_2^{i_2} + e_2^{i_2+1}\right)\right] \qquad (۱۱\text{-}۷۱)$$

$$U_{11}^{i_1 i_2} = \left[\frac{1}{2}\left(e_1^{i_1} + e_1^{i_1+1}\right), e_1^{i_1+1}\right] \times \left[\frac{1}{2}\left(e_2^{i_2} + e_2^{i_2+1}\right), e_2^{i_2+1}\right] \qquad (۱۱\text{-}۷۲)$$

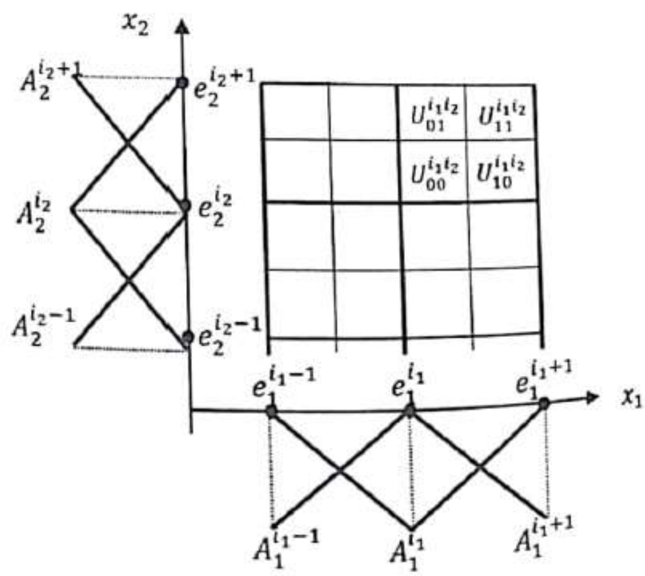
برای درک بهتر این موضوع به شکل (۱۱-۷) توجه کنید. بنابراین برای هر $x \in U$، $U_{pq}^{i_1 i_2}$ ($p,q =$ 0 یا 1) وجود دارد به نحوی که $x \in U_{pq}^{i_1 i_2}$. اگر x داخل $U_{pq}^{i_1 i_2}$ باشد آنگاه با کمک شکل (۱۱-۷) مشاهده می‌شود که $\mu_{A_1^{i_1+p}}(x_1) > 0.5$، $\mu_{A_2^{i_2+q}}(x_2) > 0.5$ و تمامی توابع تعلق دیگر کوچکتر از 0.5 خواهند بود. بنابراین از روابط (۱۱-۶۲) و (۱۱-۶۳) خواهیم داشت:

$$f(x) = g\left(e_1^{i_1+p}, e_2^{i_2+q}\right) = \frac{1}{3 + e_1^{i_1+p} + e_2^{i_2+q}} \qquad (۱۱\text{-}۷۳)$$

که از طریق دو گام زیر محاسبه می‌شود:

گام اول) برای هر $x \in U$ داده شده، p، q، i_1 و i_2 را به نحوی تعیین کنید که $x \in U_{pq}^{i_1 i_2}$

گام دوم) $f(x)$ برابر است با $g\left(e_1^{i_1+p}, e_2^{i_2+q}\right)$



شکل ۷-۱۱) تقسیم‌بندی $U^{i_1 i_2}$ به چهار زیر مجموعه

دستورات متلب مربوطه:

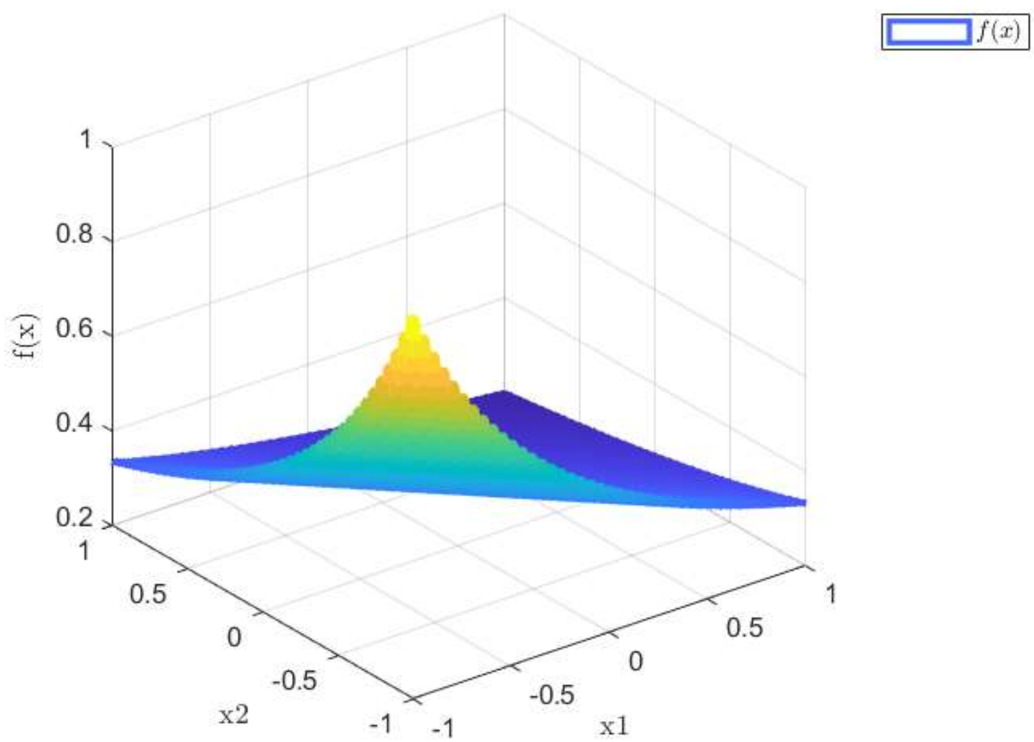
```
%% maximum defuzzifier with first order bound

clc;
clear;
```

```
close all;

tic

alfa=-1;
beta=1;
h=0.05;
N=41;

x1=alfa:0.01:beta;
x2=x1;
[~,n1]=size(x1);
[~,n2]=size(x2);

e1=beta*ones(1,N+1);
e2=beta*ones(1,N+1);

for j=1:N;
    e1(j)=alfa+h*(j-1);
    e2(j)=alfa+h*(j-1);
end

f_x=zeros(n1,n2);

for k1=1:n1
    for k2=1:n2

i1=min(find(e1<=x1(1,k1),1,'last'),find(e1>=x1(1,k1),1));

i2=min(find(e2<=x2(1,k2),1,'last'),find(e2>=x2(1,k2),1));
        if x1(1,k1)>=e1(1,i1) &&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1)) && x2(1,k2)>=e2 (1,i2)
&& x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            p=0;
            q=0;
        elseif x1(1,k1)>=e1(1,i1) &&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1)) &&
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2)) && x2(1,k2)<=e2(1,1+i2)
            p=0;
            q=1;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1)) &&
x1(1,k1)<=e1(1,1+i1) && x2(1,k2)>=e2(1,i2) &&
x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            P=1;
```

```
            q=0;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1)) &&
x1(1,k1)<=e1(1,1+i1) && x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))
&& x2(1,k2)<=e2(1,1+i2)
            p=1;
            q=1;
        end
        f_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));

    end
end
[x1,x2]=meshgrid(x1,x2);

g_x = 1./(3+x1+x2);

figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on


figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");

subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");

figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
```

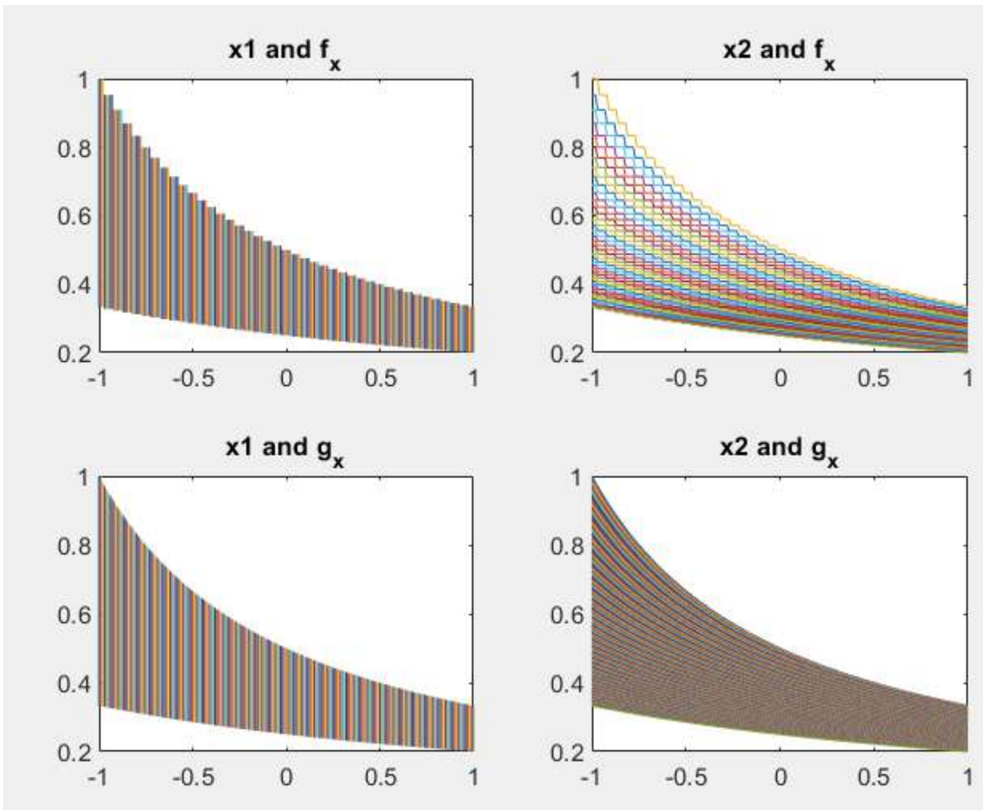
```
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");

subplot(1,2,2);
plot(x2,E);
title("x2 and Error");

toc
```

تابع f(x) تقریب‌زده شده در فضای سه بعدی بصورت زیر است:

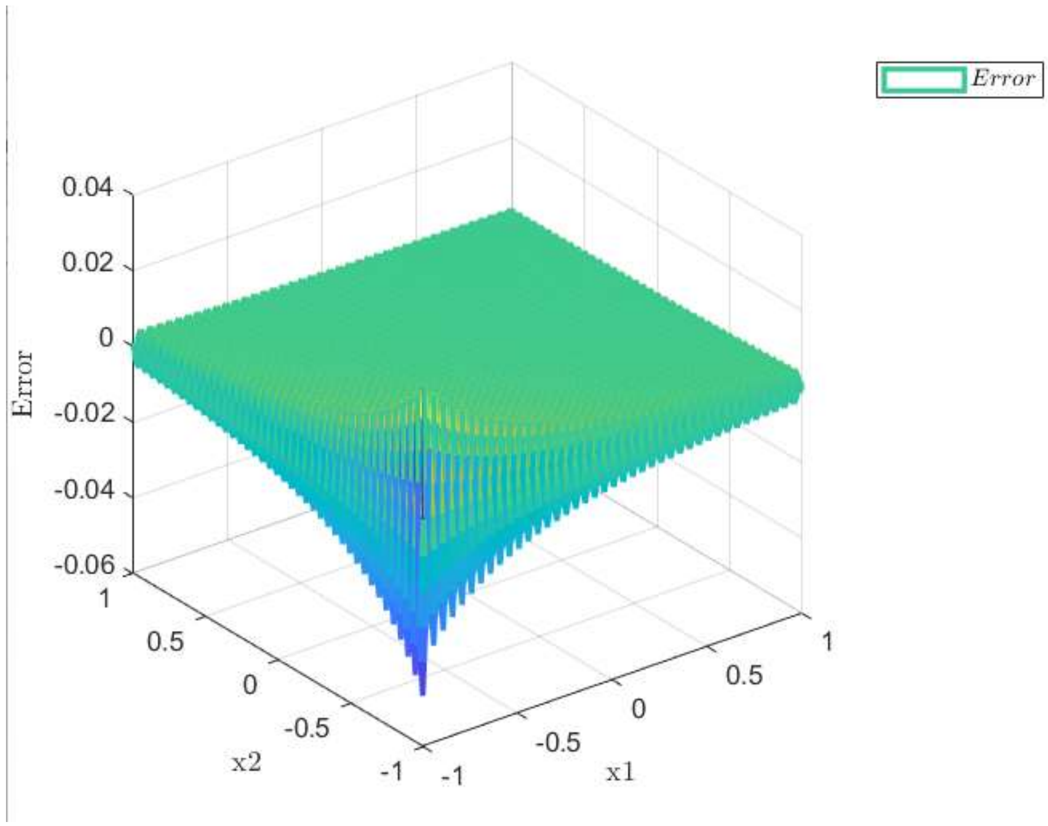


بررسی خروجی: این نمودار تابع تقریب زده شده را بر حسب x1 و x2 نشان می‌دهد. این نمودار به طور بصری نشان می‌دهد که تابع یک قله دارد، که نشان می‌دهد ممکن است در برش دوبعدی یا ممکن است یک تابع گوسی در زمینه دو متغیر x1 و x2 باشد. تابع با افزایش فاصله از قله به هر دو سمت در صفحه ایجاد شده توسط x1 و x2 کاهش می‌یابد.

رسم سیستم اصلی g(x) و سیستم تقریب‌زده‌شده f(x) در فضای ۲ بعدی بر حسب x1 و x2 :

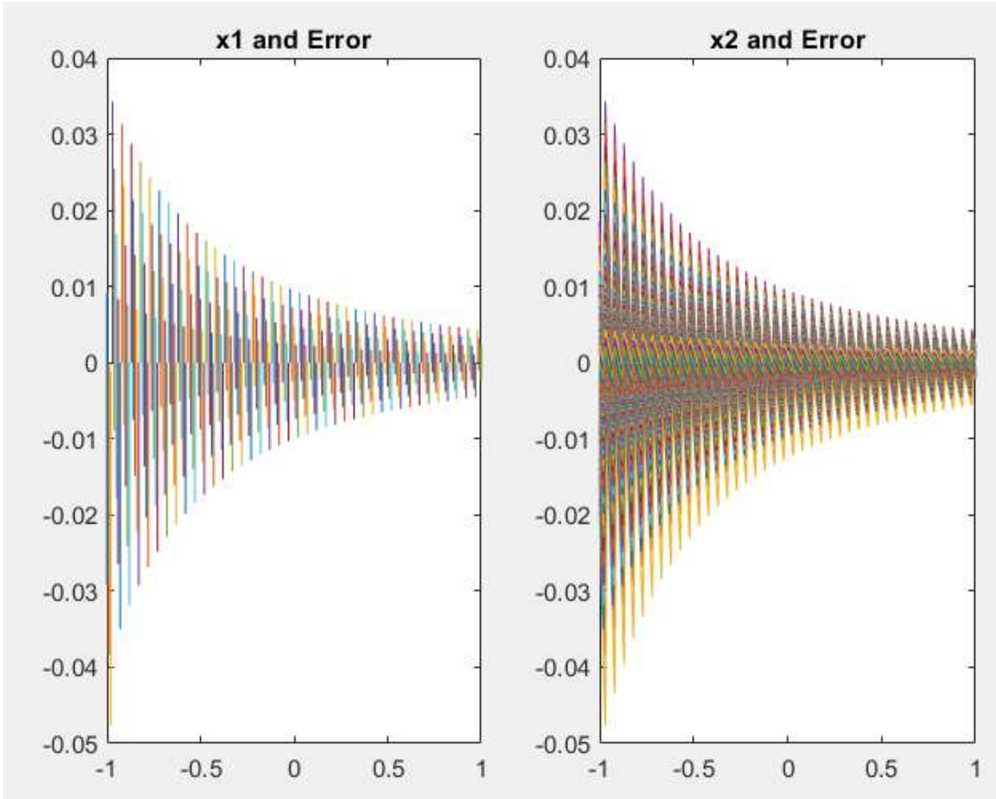


همچنین میزان خطا بین سیستم اصلی و سیستم تقریب‌زده شده بدین صورت می‌باشد:



تحلیل شکل: سطح نمودار یک شکل مخروطی با قله‌ای به سمت پایین است. این نشان می‌دهد که Error مقدار حداقلی در مرکز یا نزدیک مرکز دارد که x1 و x2 در آنجا به صفر می‌رسند. خطا هر چه دورتر از مرکز در هر جهت در صفحه x1-x2 حرکت کنیم، افزایش می‌یابد.

میزان خطا در فضای ۲ بعدی:



محاسبه مدت زمانی که طول می‌کشد تا برنامه خروجی دهد:

```
Command Window
Elapsed time is 1.046183 seconds.
fx >>
```

بخش چهارم: کران مرتبه دوم با غیرفازی ساز ماکسیمم

این قسمت مشابه قسمت قبل است با این تفاوت که مقادیر $h=0.2$ و $N=11$ می‌باشد:

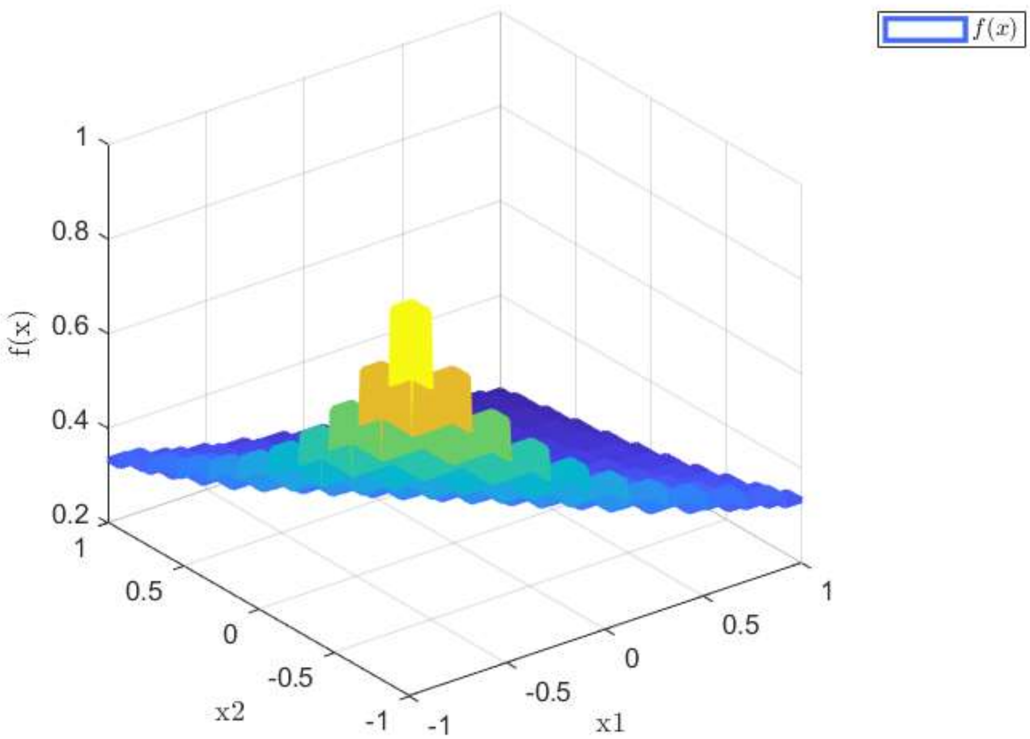
```
%% maximum defuzzifier with second order bound

clc;
clear;
close all;
tic
alfa=-1;
beta=1;
h=0.2;
N=11;
```

```
x1=alfa:0.01:beta;
x2=x1;
[~,n1]=size(x1);
[~,n2]=size(x2);

e1=beta*ones(1,N+1);
e2=beta*ones(1,N+1);

for j=1:N;
    e1(j)=alfa+h*(j-1);
    e2(j)=alfa+h*(j-1);
end

f_x=zeros(n1,n2);

for k1=1:n1
    for k2=1:n2

i1=min(find(e1<=x1(1,k1),1,'last'),find(e1>=x1(1,k1),1));

i2=min(find(e2<=x2(1,k2),1,'last'),find(e2>=x2(1,k2),1));
        if x1(1,k1)>=e1(1,i1) &&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1)) && x2(1,k2)>=e2 (1,i2)
&& x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            p=0;
            q=0;
        elseif x1(1,k1)>=e1(1,i1) &&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1)) &&
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2)) && x2(1,k2)<=e2(1,1+i2)
            p=0;
            q=1;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1)) &&
x1(1,k1)<=e1(1,1+i1) && x2(1,k2)>=e2(1,i2) &&
x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            P=1;
            q=0;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1)) &&
x1(1,k1)<=e1(1,1+i1) && x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))
&& x2(1,k2)<=e2(1,1+i2)
            p=1;
            q=1;
        end
        f_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));
```

```
    end
end

g_x = 1./(3+x1+x2);

figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on


figure(2);
subplot(2,2,1);
plot(x1,f_x);
title("x1 and f_x");

subplot(2,2,2);
plot(x2,f_x);
title("x2 and f_x");

subplot(2,2,3);
plot(x1,g_x);
title("x1 and g_x");

subplot(2,2,4);
plot(x2,g_x);
title("x2 and g_x");

figure3 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

figure(4);
subplot(1,2,1);
plot(x1,E);
title("x1 and Error");
```

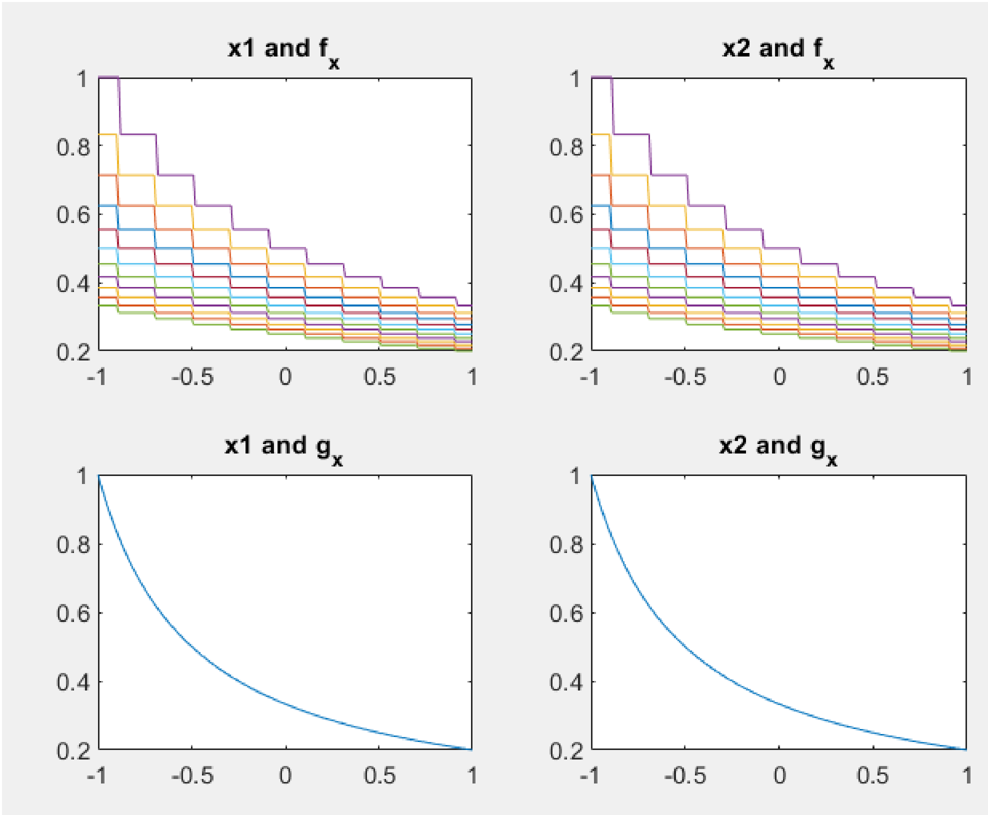
```
subplot(1,2,2);
plot(x2,E);
title("x2 and Error");

toc
```

تابع f(x) تقریب‌زده شده در فضای سه بعدی بصورت زیر است:

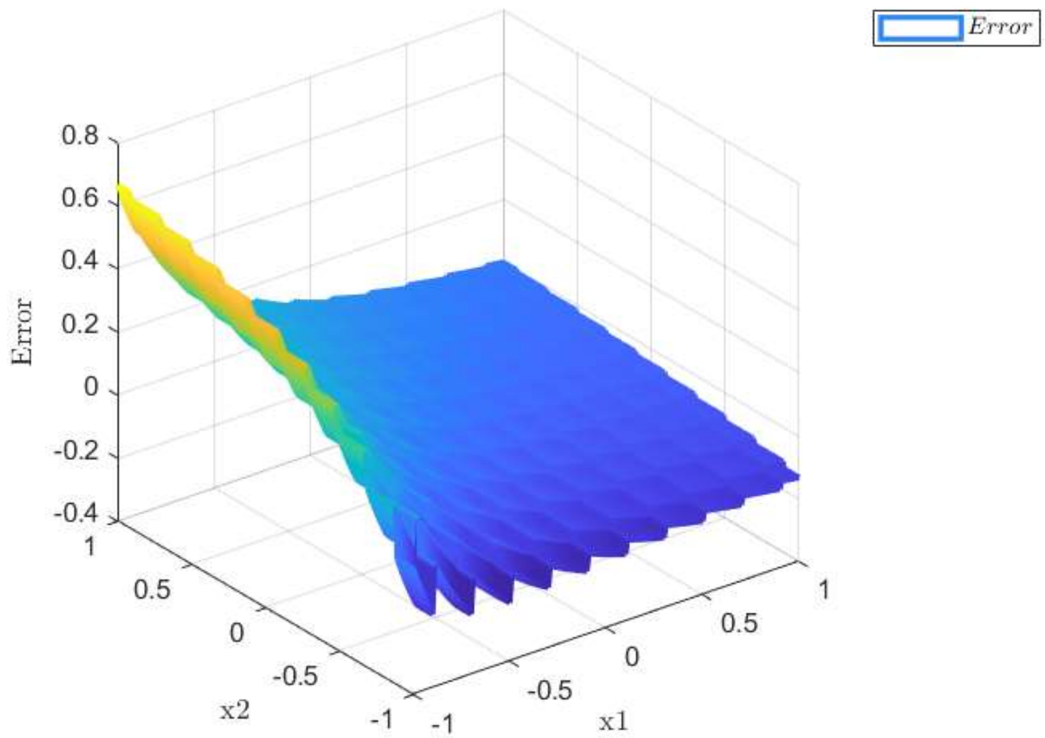


بررسی خروجی: این نمودار تابع تقریب زده شده را بر حسب x1 و x2 نشان می‌دهد. این نمودار به طور بصری نشان می‌دهد که تابع یک قله دارد، که نشان می‌دهد ممکن است در برش دوبعدی یا ممکن است یک تابع گوسی در زمینه دو متغیر x1 و x2 باشد. تابع با افزایش فاصله از قله به هر دو سمت در صفحه ایجاد شده توسط x1 و x2 کاهش می‌یابد.

رسم سیستم اصلی g(x) و سیستم تقریب‌زده‌شده f(x) در فضای ۲ بعدی بر حسب x1 و x2 :

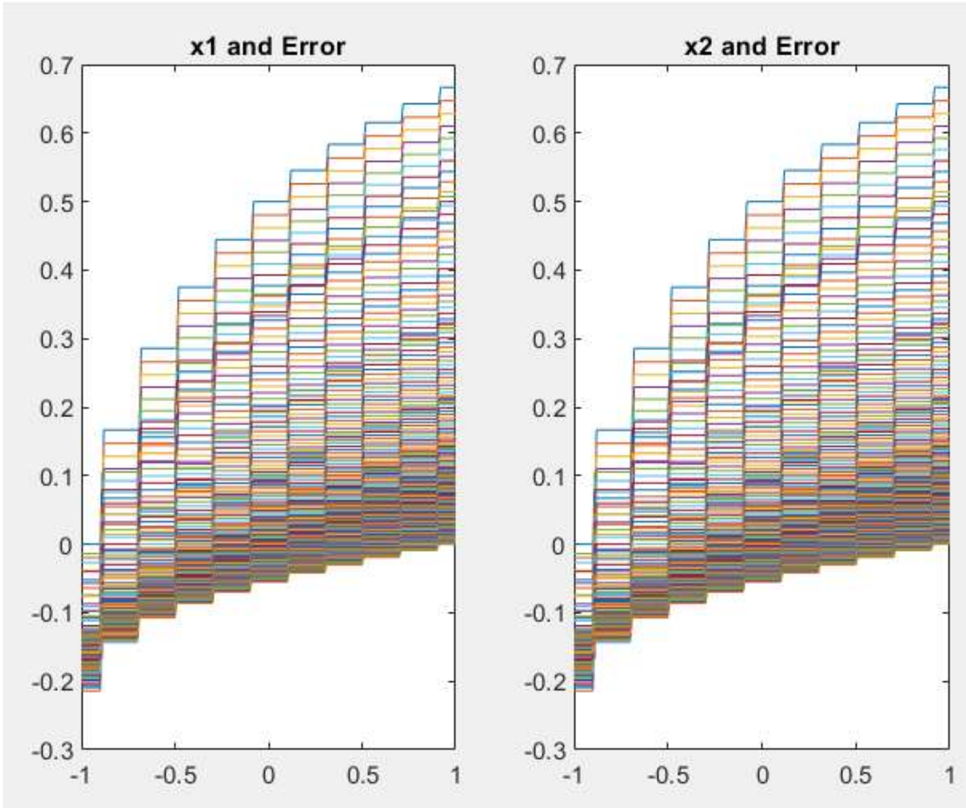


همچنین میزان خطا بین سیستم اصلی و سیستم تقریب‌زده شده بدین صورت می‌باشد:



تحلیل شکل: سطح نمودار یک شکل مخروطی با قله‌ای به سمت پایین است. این نشان می‌دهد که Error مقدار حداقلی در مرکز یا نزدیک مرکز دارد که x1 و x2 در آنجا به صفر می‌رسند. خطا هر چه دورتر از مرکز در هر جهت در صفحه x1-x2 حرکت کنیم، افزایش می‌یابد.

میزان خطا در فضای ۲ بعدی:



محاسبه مدت زمانی که طول می‌کشد تا برنامه خروجی دهد:

```
Command Window
Elapsed time is 0.766586 seconds.
fx >>
```

مقایسه کران مرتبه اول و کران مرتبه دوم:

هر دو تقریب، خروجی‌های مناسبی برای تقریب ما هستند ولی چون اولا سرعت کران مرتبه دوم بالاتر است و زودتر خروجی را به ما می‌دهد و ثانیا چون در کران مرتبه دوم از توابع تعلق کمتری استفاده کردیم، پس تقریب با کران مرتبه دوم برای ما مناسب‌تر می‌باشد.

مقایسه کران‌های مرتبه دوم غیرفازی ساز میانگین و ماکسیمم:

چون غیرفازی‌ساز ماکسیمم سریع‌تر می‌باشد پس ما این سیستم را انتخاب می‌کنیم.

سوال ۲

یک برنامهٔ کامپیوتری برای پیاده‌سازی روش جدول جستجو بنویسید. برای کامل و همه‌منظوره‌بودن برنامه، می‌توانید روش پُرکردنِ خانه‌های خالی جدول جستجو را هم در آن در نظر بگیرید. برنامهٔ خود را برای مسألهٔ پیش‌گویی سری زمانی Mackey-Glass که در بخش ۳.۱۲ مرجع [۲] آورده‌شده را به کار گرفته و اجرا کنید. نتایج را به شکلی مناسب نشان دهید.

روند حل سوال و نوشتن کدها در متلب بدین صورت می‌باشد:

۱. تنظیم پارامترها:

- کاربر دعوت می‌شود تا تعداد ورودی‌ها، تعداد توابع عضویت، نوع توابع عضویت(که برای این سوال ۳ نوع تابع توزیع مثلثی، گوسی و ذوزنقه‌ای در نظر گرفته شده که یکی را باید انتخاب کنیم)، حد پایین و حد بالا و نسبت آموزش را وارد کند.

- سپس اسکریپت ابعاد ورودی را بررسی کرده و آن‌ها را در صورت لزوم تنظیم می‌کند.

۲. نمونه‌برداری داده:

- اسکریپت مجموعه‌ای از نمونه‌های تصادفی ایجاد کرده و سپس نمونه‌برداری ثانویه را برای ایجاد مجموعه‌ای از نمونه‌ها انجام می‌دهد.

- سپس جفت‌های داده برای آموزش ایجاد می‌کند.

۳. پایگاه قوانین:

- اسکریپت از یک تابع به نام ``RuleFinder برای یافتن قوانین بر اساس داده‌های آموزش و پارامترهای توابع عضویت استفاده می‌کند.

- سپس برای یافتن قوانین تضادی بررسی می‌کند و یک پایگاه قوانین نهایی ایجاد می‌کند.

۴. سیستم فازی:

- اسکریپت با استفاده از جعبه ابزار منطق فازی در MATLAB یک سیستم فازی تعریف می‌کند.

- این سیستم ورودی و خروجی را تعریف می‌کند، توابع عضویت را برای هر متغیر تعریف می‌کند و پایگاه قوانین را به سیستم فازی اضافه می‌کند.

توضیحات توابع:

توابع RuleFinder و MFDetector به نظر می‌رسد که برای یافتن قوانین فازی و تعیین مقادیر عضویت در توابع عضویت استفاده می‌شوند. همچنین تابع ConflictChecking برای یافتن قوانین تضادی و حذف آن‌ها به کار می‌رود.

نتیجه‌گیری:

- اسکریپت از سیستم فازی برای پیش‌بینی بر اساس نمونه‌های ورودی استفاده می‌کند.

- این اسکریپت خطا را محاسبه می‌کند و مقادیر واقعی را در مقابل مقادیر پیش‌بینی شده نمایش می‌دهد.

دستورات متلب:

```
%% TABLE LOOK UP TRAINING ALGORITHEM.
clc
clear all
close all
%% 1st Part: Parameter Setting.
%%  Parameters Initiating.

disp(' Parameters Initiating...');
InpuNumb = 6;%input(' How many input do you want? Inputs
=');% Input Numbers.
disp(' ');

disp(' How many membership functions do you want?');
MemFuNu  = [18];%input(' Please Enter a vector. MFN =');
% Membership Function Numbers.
e1 = numel(MemFuNu);                    % Number of Matrix
Elements.
disp(' ');

disp(' Which membership functions type do you want?');
disp(' Triangular = 1 , Trapezoidal = 2 , Gaussian = 3');
MemFTy = [3];%input(' Please Enter a vector. MF =');
% Membership Function Types.
e4 = numel(MemFTy);                     % Number of Matrix
Elements.
disp(' ');

disp(' Enter the Lower Bound.');
LowBnd = [0.25];%input(' Please Enter a vector. Low Bnd
=');       % Lower Bound.
```

```
e2 = numel(LowBnd);                    % Number of Matrix
Elements.
disp(' ');

disp(' Enter the Upper Bound.');
% Upper Bound.
UpBnd = [1.6];%input(' Please Enter a vector. Upper Bnd
=');
e3 = numel(UpBnd);                     % Number of Matrix
Elements.

q1 = e1~=InpuNumb;
q2 = e2~=InpuNumb;
q3 = e3~=InpuNumb;
q4 = e4~=InpuNumb;
Training_Ratio = 0.5;


%%  Fixing Parameters dimentions.

switch q1
    case 1                             % Fixing Membership
Function dimentions.
        if e1==1
            MemFuNu = repmat(MemFuNu,1,InpuNumb+1);
        else
            disp(' Invalid Membership Function Number!');
            disp(' Please Start again.');
        end
end

switch q2
    case 1                             % Fixing Lower Bound
dimentions.
        if e2==1
            LowBnd = repmat(LowBnd,1,InpuNumb+1);
        else
            disp(' Invalid Lower Boundary!');
            disp(' Please Start again.');
        end
end

switch q3
```

```
    case 1                            % Fixing Upper Bound
dimentions.
        if e3==1
            UpBnd = repmat(UpBnd,1,InpuNumb+1);
        else
            disp(' Invalid Upper Boundary!');
            disp(' Please Start again.');
        end
end

switch q4
    case 1                            % Fixing Membership
Type dimentions.
        if e4==1
            MemFTy = repmat(MemFTy,1,InpuNumb+1);
        else
            disp(' Invalid Membership Function Handle!');
            disp(' Please Start again.');
        end
end
disp(' ');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-
*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-------------------------
-----------------------------')
disp(' ');
%% 2nd Part: Data.
%%  Sampling.


SAMPLES = rand(1,32);                 % Primary sampling.
%SAMPLES = 0.2:0.01:0.51;
for v=33:1033                         % Secoundary
Sampling.
    SAMPLES(v) = 0.2*SAMPLES(v-31)/(1+(SAMPLES(v-
31)^10))+0.9*SAMPLES(v-1);
end

SAMPLES = SAMPLES(34:end);            % Samples.
plot(SAMPLES)
%%  Data Paires.

e5 = numel(SAMPLES);
w1 = round(Training_Ratio*e5);
Training_Datas = zeros(w1,InpuNumb+1);
```

```
for v=1:w1
    Training_Datas(v,:) = SAMPLES(v:v+InpuNumb);  %
Creating data paires.
end

disp(' Data sampling check!');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-
*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.----------------------
-----------------------------')
disp(' ');
%% 3rd Part: Rule Base.
%%  Finding Rules.
                                                 % Implementing
a function to
                                                 % find rules.
[Rules , RulesMVlu] =
RuleFinder(Training_Datas,MemFuNu,LowBnd,UpBnd,MemFTy);
PRB = Rules(:,:);                                % Primary Rule
Base.
disp(' The Rule Base is READY!');
disp(' Here is the detailes:  ');
Rules_Number_all = numel(PRB(:,1));              % Calculating
Number of Rules.
disp(' Number of All Rules:');
disp(Rules_Number_all);

                                                 % Implimenting
a function to
Rules = ConflictChecking(Rules,RulesMVlu);  % eliminate
conflictions.
RuleBase = [Rules ones(size(Rules,1),2)];   % Creating
final Rule Base.
nonconflict_rules = numel(Rules(:,1));
disp(' And Number of No Conflicting Rules:'); %
Nonconflicting Rules.
disp(nonconflict_rules);

conflict_Rules = Rules_Number_all-nonconflict_rules; %
Calculating Number
disp(' And Number of Conflicted Rules:');            % of
Conflicted
disp(conflict_Rules);                                %
Rules.
```

```
disp(' ');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-
*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*---------------------------
----------------------------')
disp(' ');
%% 4th Part: Fuzzy System.
%%  Defining Fuzzy System.

TSP = newfis('Time Series Prediction'); % Creating the
fuzzy system.
e1  = numel(MemFuNu);

for i=1:e1-1                               % Defining the
Inputs.
    Name = ['X' num2str(i)];               % Rule Name's
Creation.
    TSP = addvar(TSP,'input',Name,[LowBnd(i),UpBnd(i)]);

    switch MemFTy(i)
        case 1                             % For Triangular
Membership Function.
            Step = (UpBnd(i)-LowBnd(i))/(MemFuNu(i)-1);
            for j = 1:MemFuNu(i)
                Center = LowBnd(i)+(j-1)*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'input',i,MemFName,'trimf',[Center-
Step,Center,Center+Step]);
            end

        case 2                            % For Trapezoidal
Membership Function.
            Step = (UpBnd(i)-LowBnd(i))/(MemFuNu(i)-1)/3;
            for j = 1:MemFuNu(i)
                Center = LowBnd(i)+(j-1)*3*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'input',i,MemFName,'trapmf',[Center-
2*Step,Center-Step,Center+Step,Center+2*Step]);
            end
```

```
        case 3                           % For Gaussian
Membership Function.
            Step = (UpBnd(i)-LowBnd(i))/(MemFuNu(i)-1)/2;
            for j = 1:MemFuNu(i)
                Center = LowBnd(i)+(j-1)*2*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'input',i,MemFName,'gaussmf',[Step,Center]);
            end
    end
end
                                         % Deffining the
Output.
TSP = addvar(TSP,'output','Y',[LowBnd(end),UpBnd(end)]);
Name = ['Y'];

switch MemFTy(end)
        case 1                           % For Triangular
Membership Function.
            Step = (UpBnd(end)-LowBnd(end))/(MemFuNu(end)-
1);
            for j=1:MemFuNu(end)
                Center = LowBnd(end)+(j-1)*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'output',1,MemFName,'trimf',[Center-
Step,Center,Center+Step]);
            end

        case 2                           % For Trapezoidal
Membership Function.
            Step = (UpBnd(end)-LowBnd(end))/(MemFuNu(end)-
1)/3;
            for j=1:MemFuNu(end)
                Center = LowBnd(end)+(j-1)*3*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'output',1,MemFName,'trapmf',[Center-
2*Step,Center-Step,Center+Step,Center+2*Step]);
            end
```

```
        case 3                         % For Gaussian
Membership Function.
            Step = (UpBnd(end)-LowBnd(end))/(MemFuNu(end)-
1)/2;
            for j=1:MemFuNu(end)
                Center = LowBnd(end)+(j-1)*2*Step;
                MemFName = [Name '_MF' num2str(j)];   %
Rule Name's Creation.
                TSP =
addmf(TSP,'output',1,MemFName,'gaussmf',[Step,Center]);
            end
end

TSP = addrule(TSP,RuleBase);          % Deploying Rule Base
to our Fuzzy System.
disp(' Fuzzy System is Ready!');
disp(' Fuzzy System detailes:');
showfis(TSP)                           % Showing Fuzzy
System detailes.

disp(' ');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-
*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*------------------------
----------------------------')
disp(' ');
%% 5th Part: Conclusion.
%%  Predicting.

Y = SAMPLES(1:InpuNumb);

for i=InpuNumb+1:numel(SAMPLES)           % Calculating
Pridictions.

    Y(i)=evalfis(SAMPLES(i-InpuNumb:i-1),TSP);

end

Error = sum(abs(SAMPLES-Y))/100;           % Calculating
Error.
ED = [' Error of Prediction = ' num2str(Error) '%' ];

%%  Plotting.

AH = axes;
```

```
sam_plt = plot(AH,SAMPLES);
hold on
output_plt = plot(AH,Y,'r');

xlim([0 e5]);
ylim([LowBnd(end)-0.2 UpBnd(end)+0.2]);
legend(AH,'Real Value','Prediction');
disp(ED);

%% Rules Finder.
%%  For implementing in Table look Up Algorithm.

function [Rule RuleMemValue] =
RuleFinder(Pair,MFNum,LowBnd,UpBnd,MemFunTyp)

e1 = numel(MFNum);
e2 = size(Pair,1);
MVal = zeros(1,e1);
MFun = zeros(1,e1);
Rule = zeros((size(Pair)));
RuleMemValue = zeros((size(Pair)));

    for i=1:e2
        for j=1:e1
            [MVal(j) MFun(j)] =
MFDetector(Pair(i,j),MFNum(j),LowBnd(j),UpBnd(j),MemFunTyp(
j));
        end

        Rule(i,:)=MFun;
        RuleMemValue(i,:)=MVal;
    end
end

%% Finding The MF Value and The MF Number of each Data.
function [M_Value MF_Num] =
MFDetector(x,Num,LowBnd,UpBnd,MemFunTyp)

Y = zeros(1,Num);

switch MemFunTyp
    case 1
        Step = (UpBnd-LowBnd)/(Num-1);
        for i=1:Num
```

```
            Center = LowBnd+(i-1)*Step;
            Y(i) = trimf(x,[Center-
Step,Center,Center+Step]);
        end

    case 2
        Step = (UpBnd-LowBnd)/(Num-1)/3;
        for i=1:Num
            Center = LowBnd+(i-1)*3*Step;
            Y(i) = trapmf(x,[Center-2*Step,Center-
Step,Center+Step,Center+2*Step]);
        end

    case 3
        Step = (UpBnd-LowBnd)/(Num-1)/2;
        for i=1:Num
            Center = LowBnd+(i-1)*2*Step;
            Y(i) = gaussmf(x,[Step,Center]);
        end

    otherwise
        disp(' Selected Membership Function is Wrong!')
        disp(' Please Start Again.')
end

[M_Value MF_Num] = max(Y);
end

%% Conflict finder and deleter.
function [Rule1 Rule1MVlu RuleDegree] =
ConflictChecking(Rule,Rule_MV)
k = 1;

    while ~isempty(Rule)
        First = repmat(Rule(1,1:end-1),size(Rule,1),1);
        Rule_B = Rule(:,1:end-1);
        Index = Rule_B == First;
        Index = find(sum(Index,2) == numel(Rule_B(1,:)));

        [RuleDegree(k) Rule_Index] =
max(prod(Rule_MV(Index,:),2));

        Rule1(k,:) = Rule(Index(Rule_Index),:);
        Rule1MVlu(k,:) = Rule_MV(Index(Rule_Index),:);
```

```
        k = k+1;
        m = 1;
        RuleBuffer = [];

        for n=1:size(Rule,1)
            if isempty(find(Index==n))
                RuleBuffer(m,:) = Rule(n,:);
                m = m+1;
            end
        end

        Rule = RuleBuffer;
    end
end
```

نتایج خروجی‌های کد:

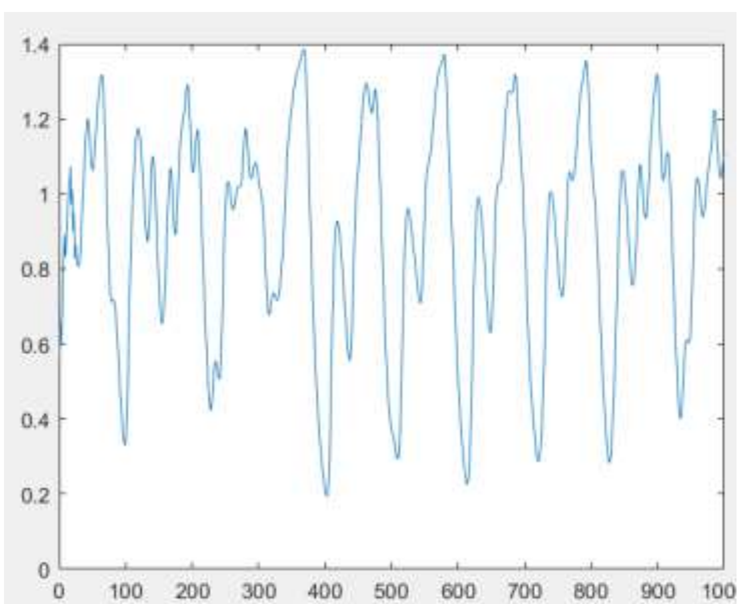
تعداد قوانین:

```
Number of All Rules:
   500
```

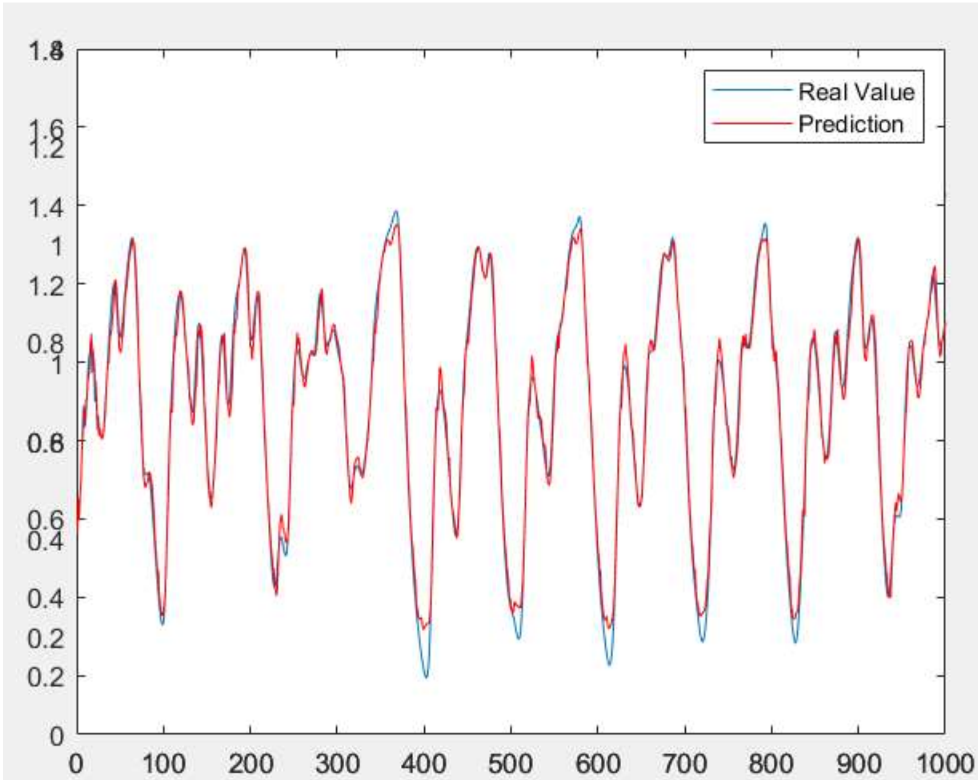
مشخصات سیستم فازی:

```
1.  Name               Time Series Prediction
2.  Type               mamdani
3.  Inputs/Outputs     [6 1]
4.  NumInputMFs        [18 18 18 18 18 18]
5.  NumOutputMFs       18
6.  NumRules           293
7.  AndMethod          min
8.  OrMethod           max
9.  ImpMethod          min
10. AggMethod          max
11. DefuzzMethod       centroid
12. InLabels           X1
13.                    X2
14.                    X3
15.                    X4
16.                    X5
17.                    X6
18. OutLabels          Y
19. InRange            [0.25 1.6]
20.                    [0.25 1.6]
21.                    [0.25 1.6]
22.                    [0.25 1.6]
23.                    [0.25 1.6]
24.                    [0.25 1.6]
25. OutRange           [0.25 1.6]
```

مشاهده سری زمانی Mackey-Glass :



مشاهده مقادیر واقعی و پیشگویی شده سری زمانی با استفاده از توابع تعلق:



مشاهده درصد خطا:

```
Error of Prediction = 0.2401%
>>
```

تحلیل نتایج: مشاهده می‌شود که سیستم فازی ما به خوبی تقریب زده و می‌بینیم که خطای آن بسیار کم است و قابل استفاده می‌باشد.

سوال ۳

فرض کنید یک سیستم با معادلهٔ دیفرانسیل آورده‌شده در معادله ۱ دارید که قرار است توسط یک شناساگر فازی شناسایی شود.

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \qquad (۱)$$

که در آن تابع نامعلوم $g[u(k)]$ براساس معادله ۲ تعریف می‌شود.

$$g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \qquad (۲)$$

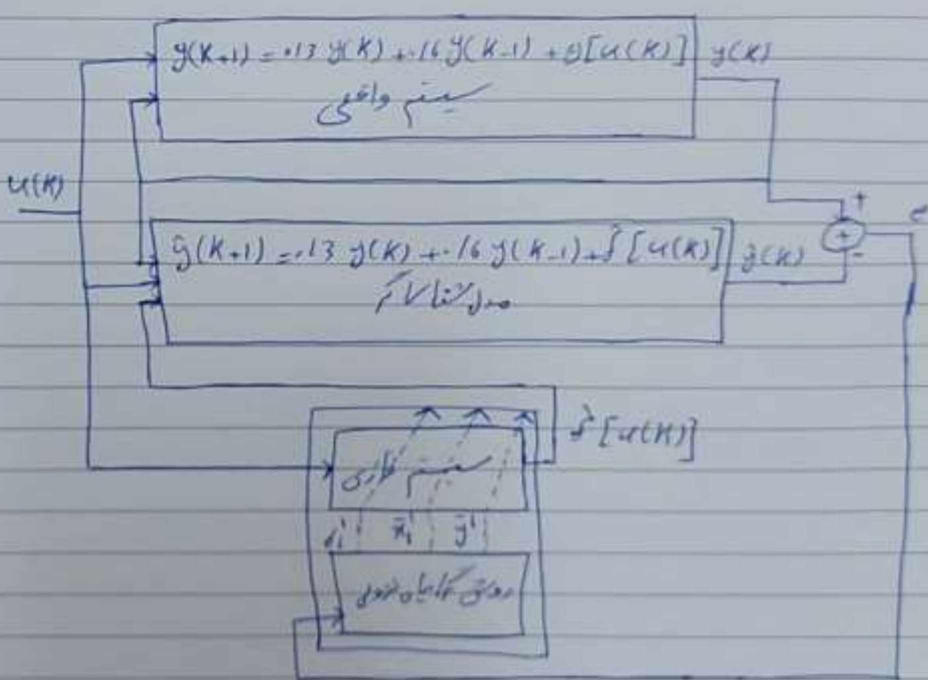
هدف ما این است که عنصر غیرخطی نامعلوم $g[u(k)]$ در معادله ۱ را توسط سیستمی فازی با رابطهٔ معادله ۳ و به‌همراه الگوریتم آموزش گرادیان نزولی (مثلاً روابط (۵.۱۳)، (۸.۱۳) و (۹.۱۳) در مرجع [۲]) تقریب بزنیم. با طراحی و برنامه‌نویسی مناسب این کار را انجام دهید.

$$f(x) = \frac{\sum_{l=1}^{M} \bar{y}^l \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right)\right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right)\right]} \qquad (۳)$$

فرض کنیم $\hat{f}(x)$ یک سیستم فازی مطابق رابطه ۳ که سوال داده است، باشد. بنابراین در این صورت در رابطه اصلی سوال $g[u(k)]$ را با $\hat{f}[u(k)]$ جایگزین کرده و مدل شناسایی زیر را به دست آوریم:

$$\hat{y}(k+1) = 0.3\, y(k) + 0.6\, y(k-1) + \hat{f}[u(k)]$$

هدف تنظیم پارامترهای $\hat{f}[u(k)]$ می‌باشد به نحوی که خروجی مدل شناسایی $\hat{y}(k+1)$ هنگامی که k به سمت بینهایت می‌رود، به خروجی سیستم واقعی $y(k+1)$ همگرا شود. طرح شناسایی بدین صورت است:



برنامه ترسیم خروجی‌های سیستم واقعی و مدل شناسایی را در زیر مشاهده می‌کنیم. در الگوریتم آموزش، λ رابرابر با ۰٫۱ در نظر می‌گیریم. مطابق رابطه ۳ در سوال، سیستم فازی را در نظر می‌گیریم و تعداد توابع تعلق را برابر $M = 5$ انتخاب می‌کنیم. پارامترهای اولیه را نیز انتخاب می‌کنیم که در کد مربوطه موجود است. بعد از انتخاب پارامترهای اولیه فرایند آموزش را به اندازه $k = 200$ مرحله زمانی وبا استفاده از ورودی تصادفی $u(k)$ که

دامنه این تابع یکنواخت در بازه [۱ ۱-] تعریف شده است؛ انجام داده و پارامترهای ، و را به کمک الگوریتم گرادیان نزولی و با استفاده از روابط زیر در هر مرحله تصحیح می‌کنیم:

$$\bar{y}^l(q+1) = \bar{y}^l(q) - \lambda \frac{f-y}{b} z^l$$

$$\bar{x}_i^l(q+1) = \bar{x}_i^l(q) - \frac{2\lambda(f-y)}{b} \frac{\left(x_{0i}^p - \bar{x}_i^l\right)}{\sigma_i^{l^2}} z^l(\bar{y}^l - f)$$

$$\sigma_i^l(q+1) = \sigma_i^l(q) - \frac{2\lambda(f-y)}{b} \frac{\left(x_{0i}^p - \bar{x}_i^l\right)^2}{\sigma_i^{l^3}} z^l\left(\bar{y}^l - f\right)$$

دستورات متلب مربوطه:

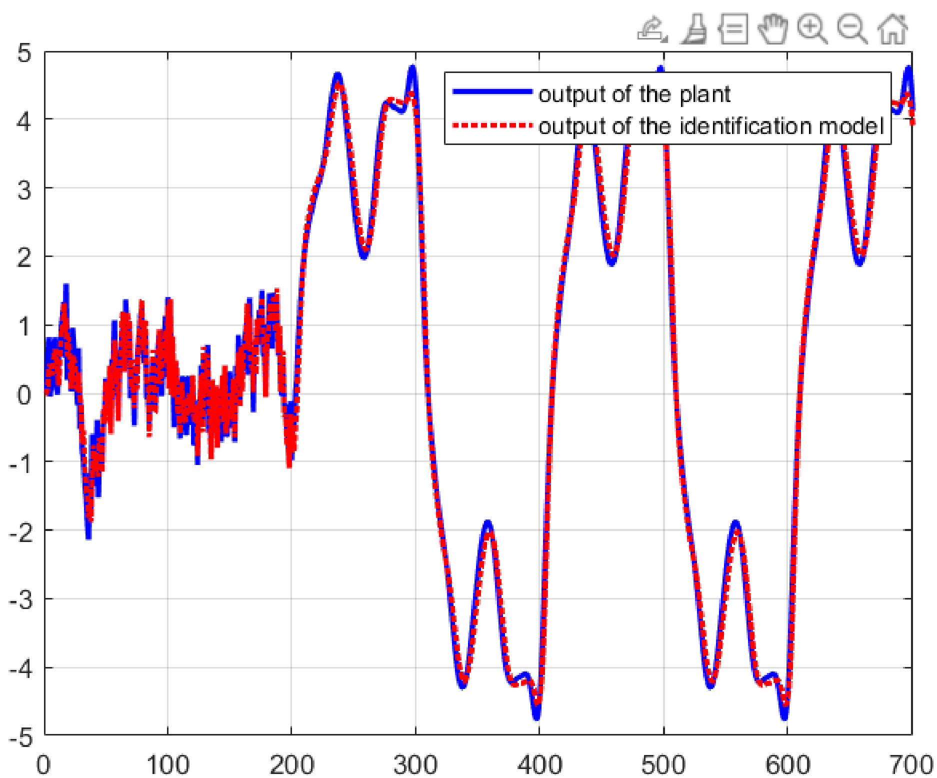
```
clear all;
clc;
close all;
%% Initializing
M=5; % Number of membership functions (Based on 1st step of
fuzzy system design)
num_training=200; % Number of training
total_num=700;
landa=0.1; % A constant stepsize
% preallocation
x_bar=zeros(num_training,M);
g_bar=zeros(num_training,M);
sigma=zeros(num_training,M);
y=zeros(total_num,1);
u=zeros(total_num,1);
x=zeros(total_num,1);
y_hat=zeros(total_num,1);
f_hat=zeros(total_num,1);
z=zeros(total_num,1);
g_u=zeros(total_num,1);
u(1)=-1+2*rand;
y(1)=0;
g_u(1)=0.6*sin(pi*u(1))+0.3*sin(3*pi*u(1))+0.1*sin(5*pi*u(1
));
f_hat(1)=g_u(1);

%% Based on the 1st step of fuzzy system design
```

```
u_min=-1;
u_max=1;
h=(u_max-u_min)/(M-1);
for k=1:M
    x_bar(1,k)=-1+h*(k-1);
    u(1,k)=x_bar(1,k);

g_bar(1,k)=0.6*sin(pi*u(1,k))+0.3*sin(3*pi*u(1,k))+0.1*sin(
5*pi*u(1,k));
end
sigma(1,1:M)=(max(u(1,:))-min(u(1,:)))/M;
x_bar(2,:)=x_bar(1,:);g_bar(2,:)=g_bar(1,:);sigma(2,:)=sigm
a(1,:);
x_bar_initial=x_bar(1,:);
sigma_initial=sigma(1,:);
y_bar_initial=g_bar(1,:);

%% Based on the 2nd and 3rd step of fuzzy system design
for q=2:num_training
    b=0;a=0;
    x(q)=-1+2*rand;
    u(q)=x(q);

g_u(q)=0.6*sin(pi*u(q))+0.3*sin(3*pi*u(q))+0.1*sin(5*pi*u(q
));
    for r=1:M
        z(r)=exp(-((x(q)-x_bar(q,r))/sigma(q,r))^2);
        b=b+z(r);a=a+g_bar(q,r)*z(r);
    end
    f_hat(q)=a/b;
    y(q+1)=0.3*y(q)+0.6*y(q-1)+g_u(q);
    y_hat(q+1)=0.3*y(q)+0.6*y(q-1)+f_hat (q);
    for r=1:M
        g_bar(q+1,r)=g_bar(q,r)-landa*(f_hat(q)-
g_u(q))*z(r)/b;
        x_bar(q+1,r)=x_bar(q,r)-landa*((f_hat(q)-
g_u(q))/b)*(g_bar(q,r)-f_hat(q))*z(r)*2*(x(q)-
x_bar(q,r))/(sigma(q,r)^2);
        sigma(q+1,r)=sigma(q,r)-landa*((f_hat(q)-
g_u(q))/b)*(g_bar(q,r)-f_hat(q))*z(r)*2*(x(q)-
x_bar(q,r))^2/(sigma(q,r)^3);
    end
end
```

```
x_bar_final=x_bar(num_training,:);
sigma_final=sigma(num_training,:);
g_bar_final=g_bar(num_training,:);

for q=num_training:700
    b=0;a=0;
    x(q)=sin(2*q*pi/200);
    u(q)=x(q) ;

g_u(q)=0.6*sin(pi*u(q))+0.3*sin(3*pi*u(q))+0.1*sin(5*pi*u(q
));
    for r=1:M
        z(r)=exp(-((x(q)-
x_bar(num_training,r))/sigma(num_training,r))^2);
        b=b+z(r);a=a+g_bar(num_training,r)*z(r);
    end
    f_hat(q)=a/b;
    y(q+1)=0.3*y(q)+0.6*y(q-1)+g_u(q);
    y_hat(q+1)=0.3*y(q)+0.6*y(q-1)+f_hat(q);
end

%% plots and Figures

figure1=figure('Color',[1 1 1]);
plot(1:701,y,'b',1:701,y_hat,'r:','Linewidth',2);
legend('output of the plant','output of the identification
model')
axis([0 701 -5 5]);
grid on

figure2=figure('Color', [1 1 1]);
xp=-2:0.001:2;
for r=1:M
    miu_x=exp(-((xp-x_bar(1,r))./(sigma(1,r))).^2);
    plot(xp,miu_x,'Linewidth',2);
    hold on
end
xlabel('u');
ylabel('initial MF''s');
axis([-1 1 0 1]);

figure3=figure('Color', [1 1 1]);
for r=1:M
```

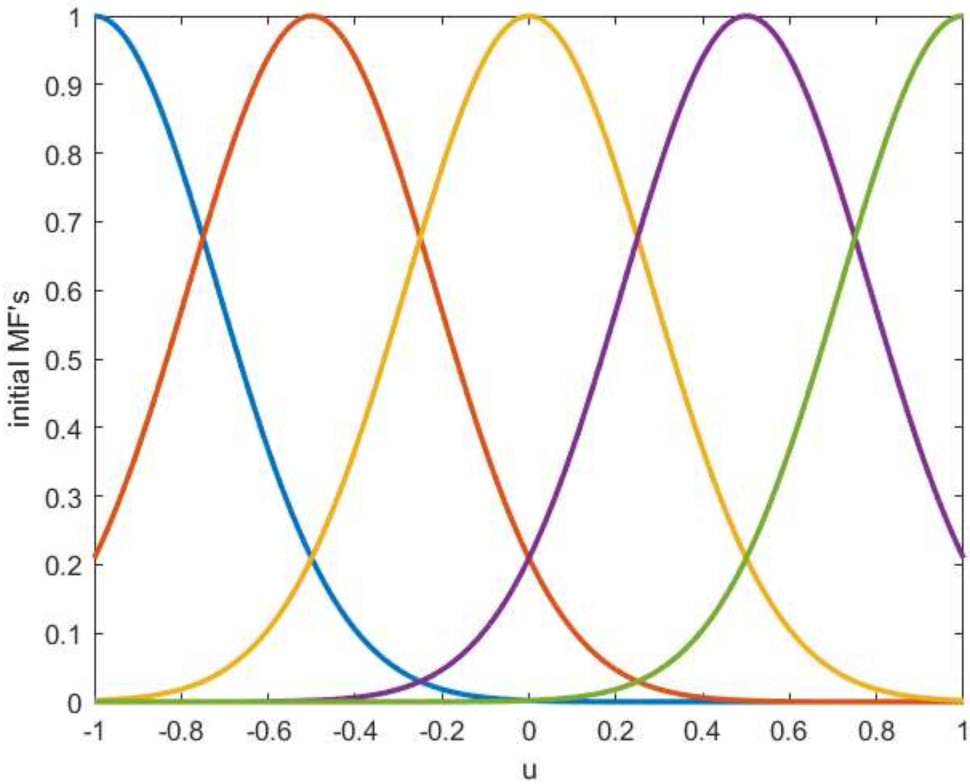
```
    miu_x=exp(-((xp-
x_bar(num_training,r))./(sigma(num_training,r))).^2);
    plot(xp,miu_x,'Linewidth',2);
    hold on
end
xlabel('u');
ylabel('final MF''s');
axis([-1 1 0 1]);
```

پاسخهای سیستم و مدل شناسایی آموزش‌یافته برای ورودی $u(k)= \sin(2\pi k/200)$ را در شکل زیر مشاهده می-کنیم:

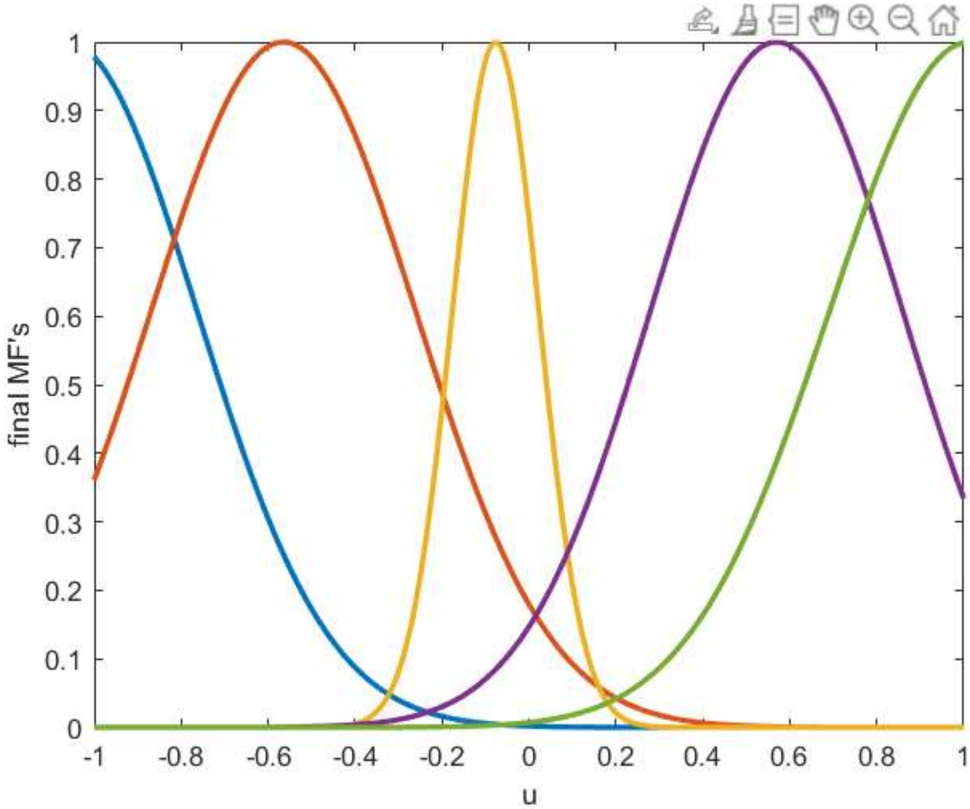


تحلیل شکل: مشاهده می‌شود که خروجی مدل شناسایی با دقت قابل قبولی خروجی سیستم را حتی پس از آنکه آموزش متوقف گردید دنبال می کند.

مشاهده توابع تعلق بر اساس پارامترهای اولیه:



مشاهده توابع تعلق نهایی:



final MF's
u
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

سوال ٤

به سوالات زیر از مبحث درخت تصمیم پاسخ دهید:

بخش اول

۱. با بهره‌گیری از آموزش ارائه‌شده در خصوص کدنویسی درخت تصمیم از ابتدا[1]، بدون استفاده از کتابخانهٔ سایکیت‌لرن دستوراتی بنویسید که درخت تصمیم یک مجموعه‌دادهٔ مربوط به بیماری کرونا که در این پیوند موجود است را خروجی دهد. اگر می‌توانید این کار را به‌صورتی انجام دهید که اطلاعات بیش‌تری را در خروجی درخت تصمیم خود دریافت کنید. لازم است که تحلیل منطقی از نتیجهٔ درخت تصمیم خود ارائه کنید. می‌توانید این کار را با الگوگرفتن از موارد گفته‌شده در ویدویهای کلاس و این پیوند انجام دهید.

توضیحات لازم برای این سوال بدین صورت می‌باشد:

۱. مقدمه:

- در این کد، از مدل درخت تصمیم (Decision Tree) برای پیش‌بینی امیدبه‌دانی استفاده شده است. ابتدا، مدل Decision Tree با استفاده از کتابخانه scikit-learn پیاده‌سازی شده است. سپس از مدل برای پیش‌بینی امیدبه‌دانی در یک مجموعه داده استفاده شده است.

۲. استفاده از کتابخانه‌ها:

- در این بخش از کتابخانه‌های pandas، numpy، matplotlib و scikit-learn استفاده شده است. این کتابخانه‌ها برای کار با داده‌ها، انجام عملیات عددی و تجسم داده‌ها و همچنین پیاده‌سازی مدل Decision Tree استفاده شده‌اند.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from graphviz import Digraph
```

۳. پیاده‌سازی مدل Decision Tree :

- در این بخش، مدل Decision Tree با استفاده از scikit-learn پیاده‌سازی شده است. ابتدا، داده‌های ورودی و خروجی برای مدل آماده شده و سپس مدل Decision Tree با استفاده از توابع مربوطه از کتابخانه scikit-learn

آموزش داده شده است. سپس مدل بر روی داده‌های تست اعمال شده و پیش‌بینی‌های مدل نمایش داده شده‌اند.

```
#https://drive.google.com/file/d/1psu-wDvYR3a0dSSs-cIsfoXI7O6CE40z/view?usp=sharing
!gdown 1psu-wDvYR3a0dSSs-cIsfoXI7O6CE40z
```

```
data = pd.read_csv('/content/covid.csv')
data
```

مشاهده خروجی:

	Fever	Cough	Breathing issues	Infected
0	No	No	No	No
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	No
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	No	No
6	Yes	No	Yes	Yes
7	Yes	No	Yes	Yes
8	No	Yes	Yes	Yes
9	Yes	Yes	No	Yes
10	No	Yes	No	No
11	No	Yes	Yes	Yes
12	No	Yes	Yes	No
13	Yes	Yes	No	No

محاسبه آنتروپی و IG :

```
labels = data['Infected']
len(labels), labels.unique(), labels.value_counts()
```

```
p = labels.value_counts() / len(labels)
-sum(p * np.log2(p))
```

مشاهده خروجی:

```
0.9852281360342515
```

```
def entropy(labels):
    p = labels.value_counts() / len(labels)
    return -sum(p * np.log2(p))

data['Infected'].value_counts()
```

مشاهده خروجی:

```
Yes    8
No     6
Name: Infected, dtype: int64
```

```
 entropy_child = 0
 for value in data['Cough'].unique():
        subset = data[data['Cough'] == value]
        print(subset)
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset['Infected'])
entropy_child
```

مشاهده خروجی:

```
   Fever Cough Breathing issues Infected
0     No    No                No       No
3    Yes    No               Yes      Yes
6    Yes    No               Yes      Yes
7    Yes    No               Yes      Yes
    Fever Cough Breathing issues Infected
1     Yes   Yes               Yes      Yes
2     Yes   Yes                No       No
4     Yes   Yes               Yes      Yes
5      No   Yes                No       No
8      No   Yes               Yes      Yes
9     Yes   Yes                No      Yes
10     No   Yes                No       No
11     No   Yes               Yes      Yes
12     No   Yes               Yes       No
13    Yes   Yes                No       No
0.9460794641311808
```

تابع نهایی آنتروپی:

```python
def entropy(labels):
    p = labels.value_counts() / len(labels)
    return -sum(p * np.log2(p))
```

```python
target = 'Infected'
entropy_parent = entropy(data[target])
entropy_parent

entropy_child = 0
feature = 'Fever'
for value in data[feature].unique():
    subset = data[data[feature] == value]
    display(subset)
    wi = len(subset) / len(data)
    entropy_child += wi * entropy(subset[target])
information_gain = entropy_parent - entropy_child

print(information_gain)
```

مشاهده خروجی:

	Fever	Cough	Breathing issues	Infected
0	No	No	No	No
5	No	Yes	No	No
8	No	Yes	Yes	Yes
10	No	Yes	No	No
11	No	Yes	Yes	Yes
12	No	Yes	Yes	No

	Fever	Cough	Breathing issues	Infected
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	No
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
6	Yes	No	Yes	Yes
7	Yes	No	Yes	Yes
9	Yes	Yes	No	Yes
13	Yes	Yes	No	No

0.12808527889139443

تابع نهایی IG :

```
def information_gain(data, feature, target):
    # Entropy of parent
    entropy_parent = entropy(data[target])

    # Entropy of child
    entropy_child = 0
    for value in data[feature].unique():
        subset = data[data[feature] == value]
        #display(subset)
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset[target])

    return entropy_parent - entropy_child
```

```python
arg=[information_gain(data, feature, 'Infected') for feature in
data.iloc[:, :-1].columns]
```

```python
def information_gain(data, feature, target):
    # Entropy of parent
    entropy_parent = entropy(data[target])

    # Entropy of child
    entropy_child = 0
    for value in data[feature].unique():
        subset = data[data[feature] == value]
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset[target])

    return entropy_parent - entropy_child
```

محاسبه IG برای فیچرهای مختلف:

```
[ ] information_gain(data, 'Fever', 'Infected')

    0.12808527889139443

[ ] information_gain(data, 'Cough', 'Infected')

    0.0391486719030707

[ ] information_gain(data, 'Breathing issues', 'Infected')

    0.39603884492804464

[ ] data.iloc[:, :-1].columns

    Index(['Fever', 'Cough', 'Breathing issues'], dtype='object')

[ ] [information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns]

    [0.12808527889139443, 0.0391486719030707, 0.39603884492804464]

[ ] np.argmax([information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns])

    2
```

مشاهده می‌کنیم که فیچر سوم بالاترین IG را دارد.

```python
class Node:
```

```
  def __init__(self, feature=None, label=None):
    self.feature = feature
    self.label = label
    self.children = {}
  def __repr__(self):
    if self.feature is not None:
      return f'DecisionNode(feature="{self.feature}",
children={self.children})'
    else:
      return f'LeafNode(label="{self.label}")'

def make_tree(data, target):
  # leaf node?
  if len(data[target].unique()) == 1:
    return Node(label=data[target].iloc[0])
  features = data.drop(target, axis=1).columns
  if len(features) == 0 or len(data) == 0:
    return Node(label=data[target].mode()[0])
  # calculate information gain
  gains = [information_gain(data, feature, target) for feature in
features]
  # greedy search to find best fearure
  max_gains_idx = np.argmax(gains)
  best_features = features[max_gains_idx]
  # make a node
  node = Node(feature=best_features)
  # loop over the best feature
  for value in data[best_features].unique():
    subset = data[data[best_features] == value].drop(best_features,
axis=1)
    # display(subset)
    node.children[value] = make_tree(subset, target)
  return node
```

این کد یک کلاس Node ایجاد می کند که برای ساخت درخت تصمیم مورد استفاده قرار می گیرد. این کلاس دارای ویژگی ها و متدهایی برای ایجاد و مدیریت گره های درخت تصمیم است. همچنین، این کد دو تابع make_tree دارد که بیان می کند که چگونه درخت تصمیم از داده ها و ویژگی های مختلف ایجاد می شود.

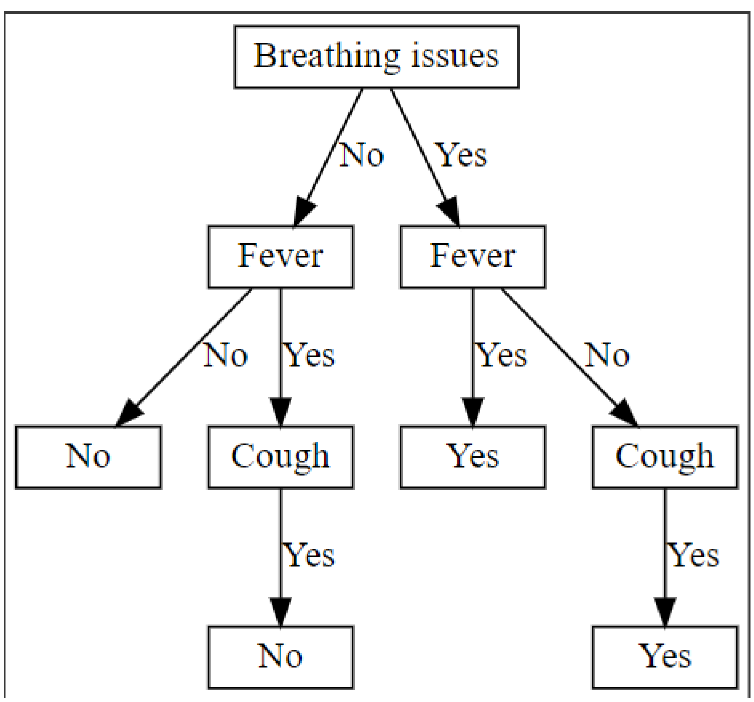
```
tree = make_tree(data, 'Infected')
tree
```

مشاهده خروجی:

```
DecisionNode(feature="Breathing issues", children={'No': DecisionNode(feature="Fever", children={'No':
LeafNode(label="No"), 'Yes': DecisionNode(feature="Cough", children={'Yes': LeafNode(label="No")})}),
'Yes': DecisionNode(feature="Fever", children={'Yes': LeafNode(label="Yes"), 'No':
DecisionNode(feature="Cough", children={'Yes': LeafNode(label="Yes")})})})
```

```python
def visualize_tree(tree, parent=None, node_id=None):
    if node_id is None:
        node_id = '0'
        g = Digraph(node_attr={'shape': 'record', 'height':'.1'})
        g.node(node_id, label=tree.feature)
    else:
        g = parent
        g.node(node_id, label=tree.feature)
    if len(tree.children) == 0:
        g.node(node_id, label=tree.label)
        return g
    for i, (value, child) in enumerate(tree.children.items()):
        child_id = f'{node_id}_{i+1}'
        visualize_tree(child, g, child_id)
        g.edge(node_id, child_id, label=value)
    return g
g = visualize_tree(tree)
g.render('decision_tree', format='png', view=True)
visualize_tree(tree)
```

مشاهده خروجی و درخت تصمیم نهایی:



بخش دوم

۲. به انتخاب خود یکی از دو مجموعه‌دادهٔ load_breast_cancer و Drugs[۲] را انتخاب کنید و کار طبقه‌بندی با درخت تصمیم را با استفاده از دستوراتی که آموزش دیده‌اید (کدنویسی از ابتدا و یا کدنویسی با کمک کتابخانهٔ سایکیت‌لرن) انجام دهید. لازم است که توضیحات مختصری از مجموعه‌داده و منطق درخت تصمیم تولیدشده بنویسید. منطق معیاری که استفاده می‌کنید و نتایج آن در قسمت‌های مختلف را به‌صورت کامل تحلیل کنید. همچنین، مسیر مربوط به دو نمونه از داده‌های مجموعهٔ آزمون را نشان داده و تحلیل کنید. اگر از فراپارامتر خاصی مانند فراپارامترهای مخصوص هرس‌کردن استفاده می‌کنید لازم است که حداقل دو مقدار بزرگ و کوچک برای آن در نظر بگیرید و تحلیل خود از تأثیر آن روی نتیجهٔ نهایی را بنویسید.

داده اولی را برای این سوال انتخاب می‌کنیم:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
# Example with max_depth as a pruning parameter
max_depth_values = [5 ,10]  # Replace with your desired values
for max_depth in max_depth_values:
    clf = DecisionTreeClassifier(max_depth=max_depth)

    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - Max Depth: {max_depth}')
    plt.savefig(f'decision_tree_max_depth_{max_depth}.png')
```

```
plt.show()
```

توضیحات کد:

این کد برای آموزش یک مدل تصمیم‌گیری درختی بر روی مجموعه داده سرطان پستان و نمایش درخت تصمیم استفاده می‌شود.

۱. وارد کردن کتابخانه‌های مورد نیاز:

- `: `from sklearn.datasets import load_breast_cancer`وارد کردن مجموعه داده سرطان پستان از کتابخانه scikit-learn.

- `: `from sklearn.model_selection import train_test_split`وارد کردن تابع تقسیم مجموعه داده به داده‌های آموزش و آزمون.

- `: `from sklearn.tree import DecisionTreeClassifier, plot_tree`وارد کردن مدل تصمیم‌گیری درختی و تابع نمایش درخت تصمیم.

- `: `import matplotlib.pyplot as plt`وارد کردن کتابخانه برای نمایش نمودارها.

۲. بارگذاری مجموعه داده سرطان پستان:

- `: `data = load_breast_cancer()`بارگذاری مجموعه داده سرطان پستان.

- `: `X = data.data`استخراج ویژگی‌ها از مجموعه داده.

- `: `y = data.target`استخراج متغیر هدف (برچسب‌ها) از مجموعه داده.

۳. تقسیم مجموعه داده به داده‌های آموزش و آزمون:

- `: `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=83)` تقسیم مجموعه داده به ۸۰% داده‌های آموزش و ۲۰% داده‌های آزمون.

٤. ساخت مدل تصمیم‌گیری درختی و نمایش درخت تصمیم:

- حلقه از مقادیر مختلف `max_depth`برای درخت تصمیم:

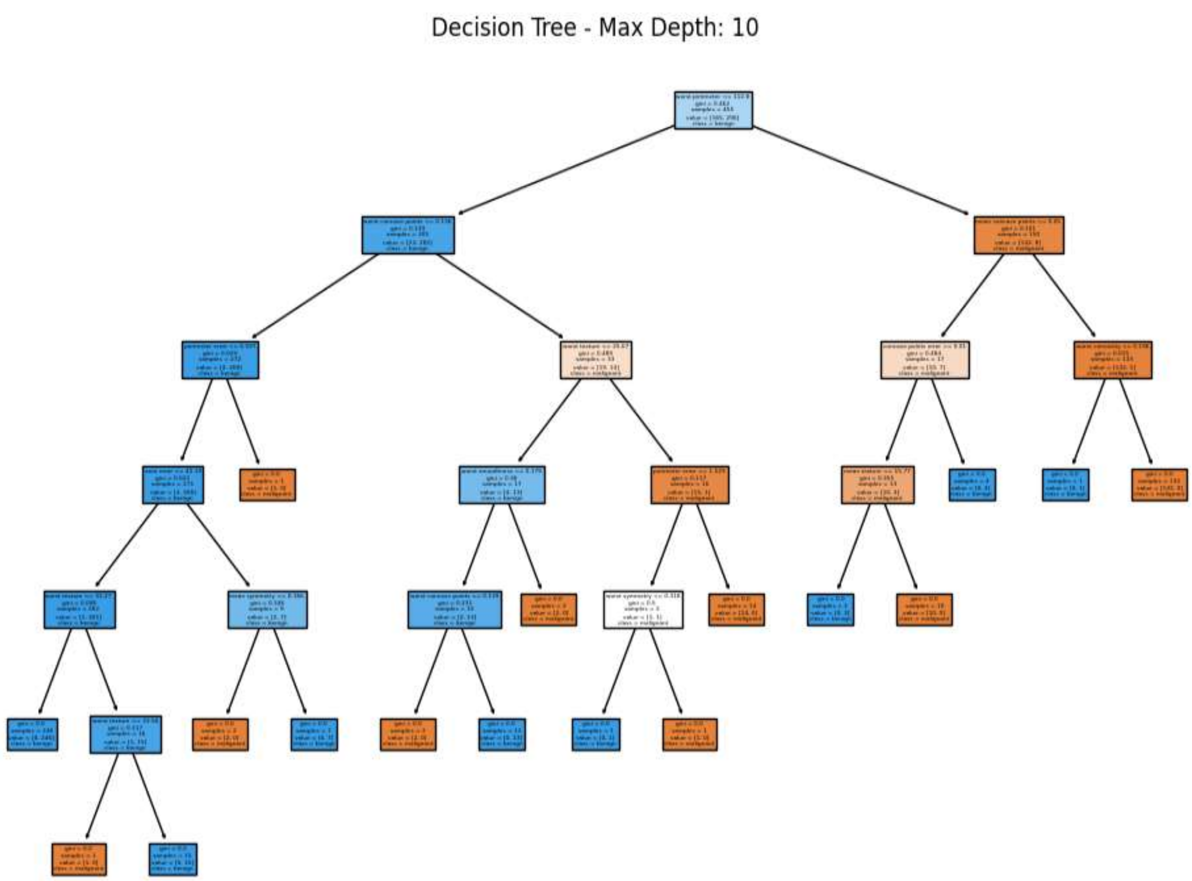
- `clf = DecisionTreeClassifier(max_depth=max_depth)`: ساخت یک مدل تصمیم‌گیری درختی با یک مقدار خاص برای `max_depth`.

- `clf.fit(X_train, y_train)`: آموزش مدل تصمیم‌گیری درختی بر روی داده‌های آموزش.

- نمایش درخت تصمیم با استفاده از `plot_tree` و ذخیره تصویر به عنوان یک فایل تصویر.

این کد به ما امکان می‌دهد با مقادیر مختلف برای max_depth برای درخت تصمیم آزمایش کنیم و درخت‌های تصمیم حاصل را برای هر مقدار max_depth مشاهده کنیم.

مشاهده خروجی:




```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
```

```
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
# Example with ccp_alpha as a pruning parameter
ccp_alpha_values = [0.0, 0.01, 0.02]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```

توضیحات کد:

این کد Python برای استفاده از کتابخانه scikit-learn برای آموزش یک مدل تصمیم‌گیری درختی بر روی مجموعه داده سرطان پستان و نمایش درخت تصمیم استفاده می‌شود. همچنین شامل آزمایش با مقادیر مختلف برای پارامتر ``ccp_alpha``می‌شود که یک پارامتر مربوط به اصلاح درخت تصمیم است و برای کنترل پیچیدگی درخت تصمیم استفاده می‌شود.

۱. **بارگیری مجموعه داده:**

- مجموعه داده سرطان پستان با استفاده از تابع ``load_breast_cancer()``از scikit-learn بارگیری می‌شود. ویژگی‌ها در ``X``و متغیر هدف در ``y``ذخیره می‌شود.

۲. **تقسیم مجموعه داده:**

- مجموعه داده به دو بخش آموزش و آزمون تقسیم می‌شود با استفاده از تابع ``train_test_split()``، ۸۰% از داده برای آموزش و ۲۰% برای آزمون استفاده می‌شود.

۳. **ایجاد یک مدل تصمیم‌گیری درختی:**

- یک مدل تصمیم‌گیری درختی با استفاده از ``DecisionTreeClassifier()``ایجاد می‌شود.

٤. **آزمایش با مقادیر مختلف `ccp_alpha`:**

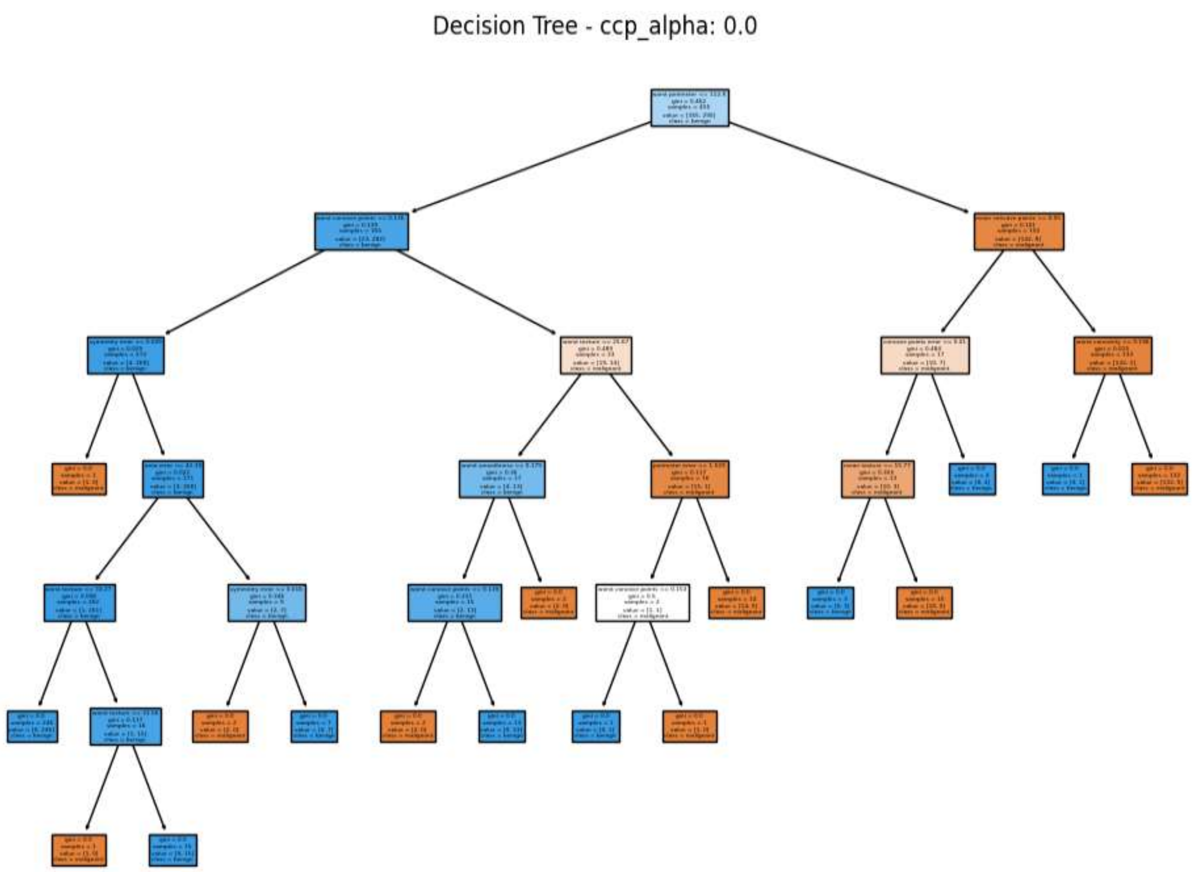
- یک حلقه بر روی مقادیر مختلف `ccp_alpha` (پارامتر اصلاح) ایجاد می‌شود.

- برای هر مقدار `ccp_alpha`، مدل تصمیم‌گیری درختی بر روی داده‌های آموزش آموزش داده می‌شود.

- درخت تصمیم نمایش داده شده و به عنوان یک فایل تصویر ذخیره می‌شود با استفاده از توابع ``plot_tree()``و ``savefig()``از matplotlib.

این کد به ما امکان می‌دهد تا درخت تصمیم را برای مقادیر مختلف ``ccp_alpha``مشاهده کرده و مشاهده کنیم که چگونه اصلاح تأثیر ساختار درخت تصمیم را تغییر می‌دهد.

مشاهده خروجی:




```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
```

```python
# Example with ccp_alpha as a pruning parameter
ccp_alpha_values = [0.0, 0.01, 0.02]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```

توضیحات کد:

۱. **بارگذاری و آماده‌سازی داده:**

- مجموعه داده سرطان پستان با استفاده از تابع ``load_breast_cancer()``بارگذاری می‌شود. داده‌های ویژگی در متغیر ``X``و داده‌های هدف در متغیر ``y``ذخیره می‌شوند.

۲. **تقسیم مجموعه داده:**

- مجموعه داده به دو بخش آموزش و آزمون با استفاده از تابع ``train_test_split()``تقسیم می‌شود. ۸۰٪ داده برای آموزش (``X_train`` و `y_train`) و ۲۰٪ برای آزمون (``X_test`` و `y_test`)استفاده می‌شود.

۳. **ایجاد یک مدل تصمیم‌گیری درختی:**

- یک مدل تصمیم‌گیری درختی با استفاده از تابع ``DecisionTreeClassifier()``ایجاد می‌شود. کد پیشنهاد می‌دهد که با هایپرپارامترهای مختلف، از جمله پارامترهای مربوط به اصلاح ((pruning، آزمایش کنید. به عنوان مثال، از ``ccp_alpha``به عنوان یک پارامتر اصلاحی استفاده شده است.

٤. **حلقه بر روی مقادیر مختلف `ccp_alpha`:**

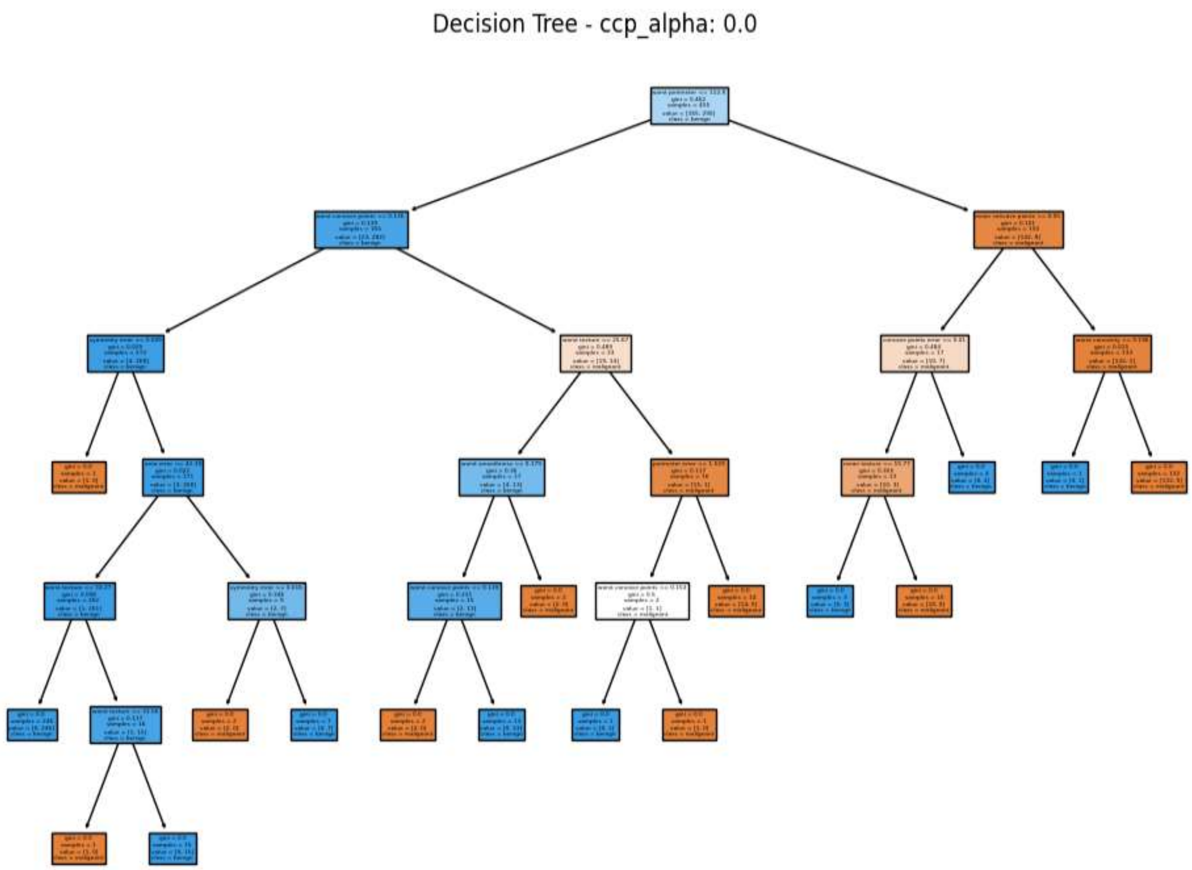
- کد بر روی مقادیر مختلف `ccp_alpha` (`0.0`، ۰٫۰۱، ۰٫۰۲ در این مثال) حرکت می‌کند.

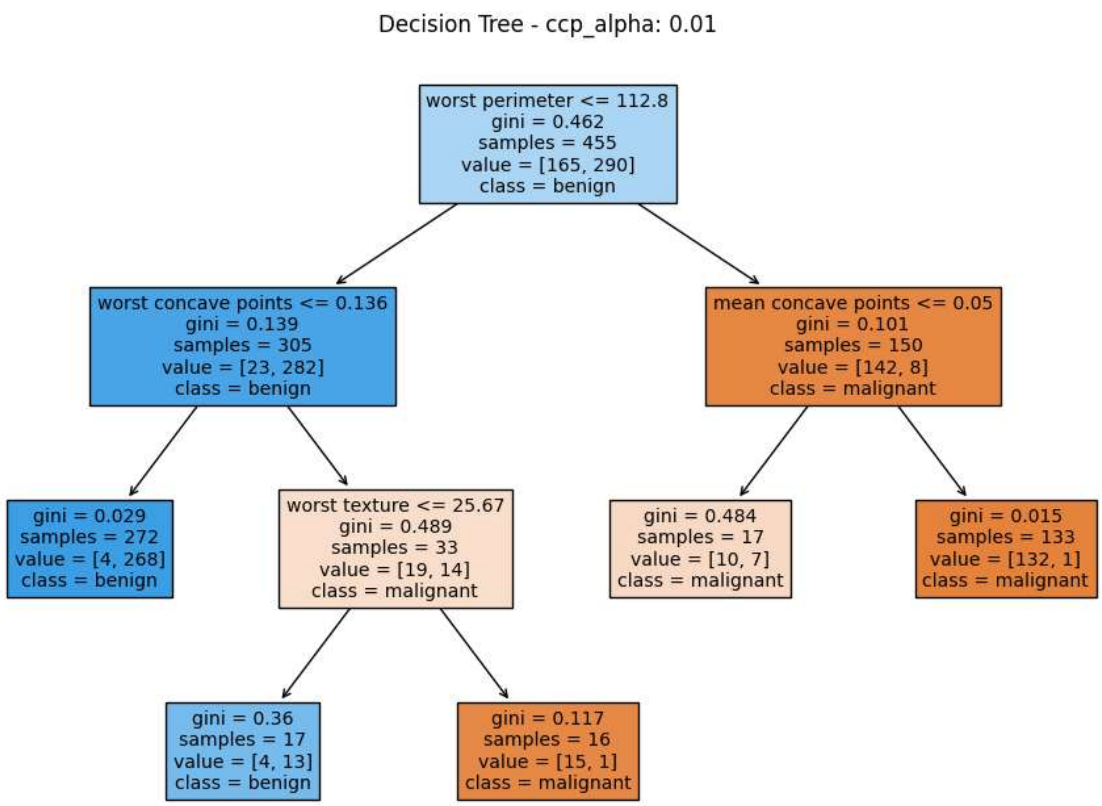
- برای هر مقدار `ccp_alpha`، یک مدل تصمیم‌گیری درختی روی داده‌های آموزش آموزش داده می‌شود.

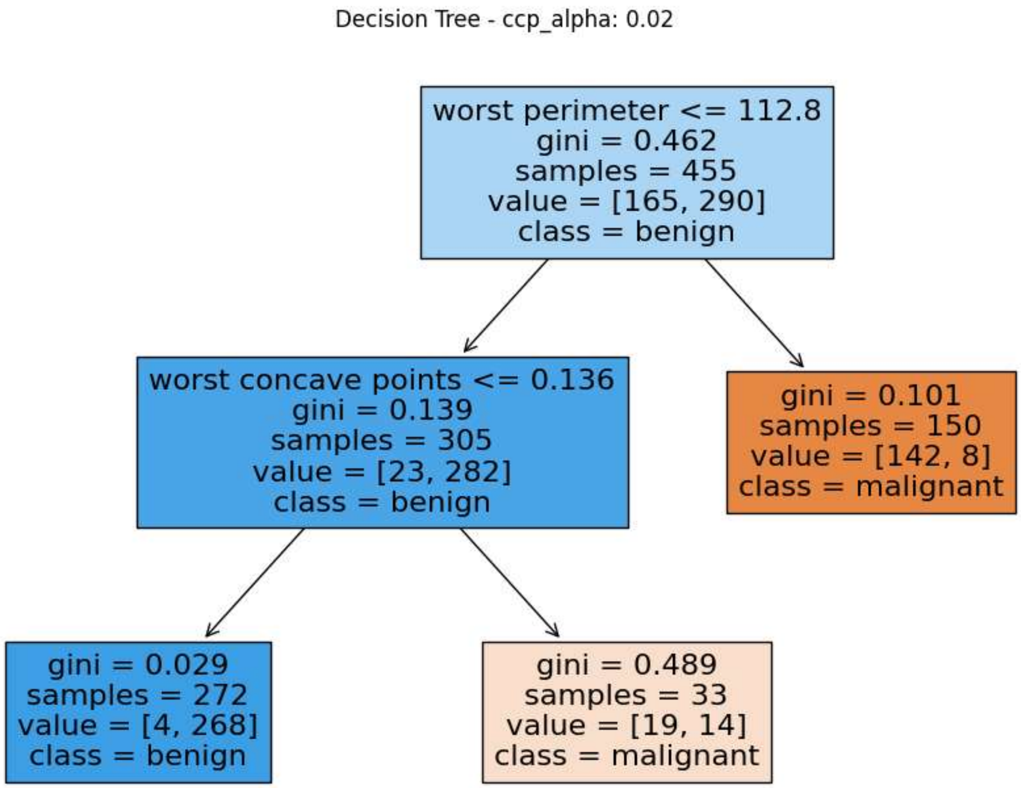
– سپس درخت تصمیم رسم شده و به عنوان یک فایل تصویری با استفاده از توابع ``plot_tree()`` و ``savefig()`` از کتابخانه ``matplotlib`` ذخیره می‌شود.

این کد به ما امکان می‌دهد تا ساختار درخت تصمیم را برای مقادیر مختلف ``ccp_alpha`` مشاهده کرده و مشاهده کنیم که چگونه اصلاح تأثیر ساختار درخت تصمیم را تغییر می‌دهد.

مشاهده خروجی:




Decision Tree - ccp_alpha: 0.01
worst perimeter <= 112.8
gini = 0.462
samples = 455
value = [165, 290]
class = benign
worst concave points <= 0.136
gini = 0.139
samples = 305
value = [23, 282]
class = benign
mean concave points <= 0.05
gini = 0.101
samples = 150
value = [142, 8]
class = malignant
gini = 0.029
samples = 272
value = [4, 268]
class = benign
worst texture <= 25.67
gini = 0.489
samples = 33
value = [19, 14]
class = malignant
gini = 0.484
samples = 17
value = [10, 7]
class = malignant
gini = 0.015
samples = 133
value = [132, 1]
class = malignant
gini = 0.36
samples = 17
value = [4, 13]
class = benign
gini = 0.117
samples = 16
value = [15, 1]
class = malignant




```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
```

```
# Example with ccp_alpha as a pruning parameter
ccp_alpha_values = [0.5]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```

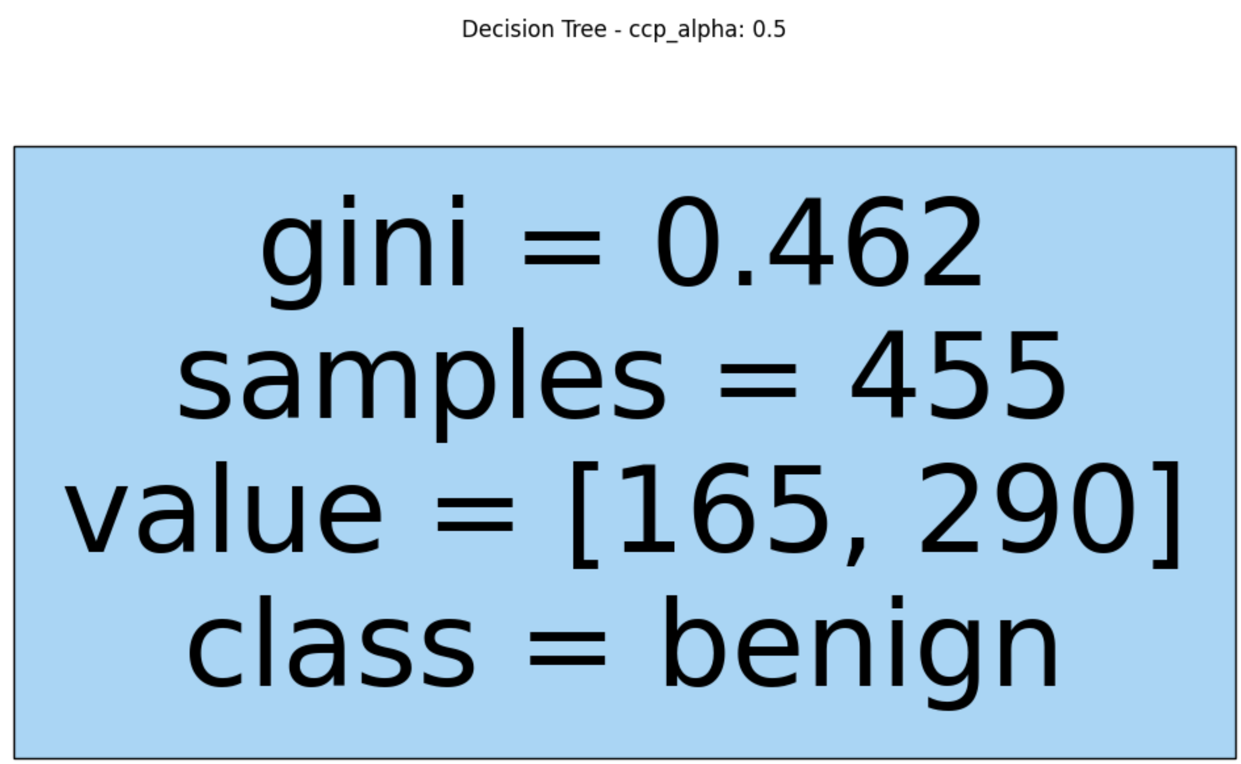
توضیحات کد:

این کد یک مدل درخت تصمیمی را برای مجموعه داده سرطان پستان ایجاد می‌کند و سپس ساختار آن را به صورت تصویری نمایش می‌دهد. این کد به زبان پایتون نوشته شده است و از کتابخانه‌های scikit-learn و matplotlib برای ایجاد مدل و تصویرسازی استفاده می‌کند.

در این کد، ابتدا مجموعه داده سرطان پستان با استفاده از تابع ``load_breast_cancer()``بارگذاری می‌شود و سپس به دو بخش آموزش و آزمون تقسیم می‌شود. سپس یک مدل درخت تصمیمی با استفاده از تابع ``DecisionTreeClassifier()``ایجاد می‌شود و روی داده‌های آموزش آموزش داده می‌شود.

سپس با استفاده از تابع ``plot_tree()``، ساختار درخت تصمیم رسم می‌شود و به عنوان یک تصویر با استفاده از تابع‌های ``plt.figure()`, `plt.title()`, `plt.savefig()`و ``plt.show()``ذخیره و نمایش داده می‌شود. همچنین، می‌توانید با تغییر مقادیر ``ccp_alpha``و تنظیمات دیگر، از جمله اضافه کردن توضیحات به نودهای درخت، به تنظیمات دلخواه خود برای رسم درخت تصمیمی بپردازید.

مشاهده خروجی:




```python
# Analyze two samples from the test set
sample1 = X_test[0]
sample2 = X_test[1]

# Make predictions for the samples
prediction1 = clf.predict([sample1])[0]
prediction2 = clf.predict([sample2])[0]

# Display the results
print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")

# Sample 1
print("Sample 1:")
print("Features:", sample1)
print("True Label:", y_test[0])
print("Predicted Label:", prediction1)
print("\n")
```

```
    # Sample 2
    print("Sample 2:")
    print("Features:", sample2)
    print("True Label:", y_test[1])
    print("Predicted Label:", prediction2)
```

در این قسمت از کد، دو نمونه از مجموعه داده آزمون را انتخاب می‌کنیم و سپس پیش‌بینی‌های مدل را برای این دو نمونه انجام می‌دهیم. سپس نتایج را نمایش می‌دهیم.

در این قسمت، ابتدا دو نمونه از مجموعه داده آزمون با استفاده از `X_test` انتخاب می‌شود. سپس با استفاده از `clf.predict()`، پیش‌بینی برچسب کلاس برای هر یک از این نمونه‌ها انجام می‌شود. سپس نتایج به صورت متنی نمایش داده می‌شود.

در نمونه اول، ویژگی‌های نمونه واقعی و برچسب واقعی و پیش‌بینی شده نمایش داده می‌شود. این عملیات برای نمونه دوم نیز انجام می‌شود.

در نهایت، با استفاده از `f-string`، یک پیام با عنوان "Analysis for Decision Tree with max_depth={max_depth}" و نتایج برای هر یک از نمونه‌ها نمایش داده می‌شود.

مشاهده خروجی:

```
Analysis for Decision Tree with max_depth=10:

Sample 1:
Features: [1.422e+01 2.312e+01 9.437e+01 6.099e+02 1.075e-01 2.413e-01 1.981e-01
 6.618e-02 2.384e-01 7.542e-02 2.860e-01 2.110e+00 2.112e+00 3.172e+01
 7.970e-03 1.354e-01 1.166e-01 1.666e-02 5.113e-02 1.172e-02 1.574e+01
 3.718e+01 1.064e+02 7.624e+02 1.533e-01 9.327e-01 8.488e-01 1.772e-01
 5.166e-01 1.446e-01]
True Label: 0
Predicted Label: 1


Sample 2:
Features: [1.747e+01 2.468e+01 1.161e+02 9.846e+02 1.049e-01 1.603e-01 2.159e-01
 1.043e-01 1.538e-01 6.365e-02 1.088e+00 1.410e+00 7.337e+00 1.223e+02
 6.174e-03 3.634e-02 4.644e-02 1.569e-02 1.145e-02 5.120e-03 2.314e+01
 3.233e+01 1.553e+02 1.660e+03 1.376e-01 3.830e-01 4.890e-01 1.721e-01
 2.160e-01 9.300e-02]
True Label: 0
Predicted Label: 1
```

```
    # Make predictions on the test set
```

```
    y_pred = clf.predict(X_test)

    # Calculate accuracy
    accuracy = metrics.accuracy_score(y_test, y_pred)

    # Display the results
    print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")
    print("Accuracy:", accuracy)
```

در این قسمت از کد، پیش‌بینی‌های مدل بر روی مجموعه داده آزمون انجام می‌شود و سپس دقت مدل محاسبه می‌شود. نتایج در ادامه نمایش داده می‌شود.

در این بخش، ابتدا با استفاده از "clf.predict(X_test)، برچسب‌های پیش‌بینی شده برای مجموعه داده آزمون محاسبه می‌شود. سپس با استفاده از "metrics.accuracy_score(y_test, y_pred)، دقت پیش‌بینی‌ها محاسبه می‌شود.

در نهایت، با استفاده از "f-string، یک پیام با عنوان " Analysis for Decision Tree with max_depth={max_depth}" و دقت مدل نمایش داده می‌شود.

مشاهده خروجی:

```
Analysis for Decision Tree with max_depth=10:

Accuracy: 0.5877192982456141
```

```
# Set max_depth to 5
max_depth = 5

# Create a decision tree classifier
clf = DecisionTreeClassifier(max_depth=max_depth)

# Train the model
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)
```

```
# Display the results
print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")
print("Accuracy:", accuracy)
```

کد بالا، یک مدل درخت تصمیم با عمق حداکثر ۵ را ایجاد می‌کند و سپس این مدل را با استفاده از داده‌های آموزشی آموزش می‌دهد. سپس مدل را بر روی داده‌های آزمون ارزیابی می‌کند و دقت آن را محاسبه می‌کند. در نهایت، نتایج ارزیابی نمایش داده می‌شود.

مشاهده خروجی:

```
Analysis for Decision Tree with max_depth=5:

Accuracy: 0.9210526315789473
```

بخش سوم

۳. سوال اختیاری: مجموعه‌دادهٔ مربوط به «میزان امید به زندگی» که در این پیوند آورده شده را فراخوانی کنید و توضیحاتی در مورد آن بنویسید. در ادامه، از دستورات مربوط به درخت تصمیم استفاده کنید و نشان دهید که با تنظیم مناسب پارامترها می‌توان پیش‌بینی مربوط به این دیتاست را روی یک مجموعهٔ آزمون به خوبی انجام داد.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
```

```
!pip install --upgrade --no-cache-dir gdown
!gdown 13UXkURa_S_QaHNsBO0m1UzbrVH1cakJK
```

```
data = pd.read_csv('/content/Life Expectancy Data.csv')
data
```

```
from sklearn.preprocessing import LabelEncoder
```

```python
categorical_columns = ['Country', 'Status']

label_encoder = LabelEncoder()

for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])
data = data.dropna()
data
```

این کد از کلاس ``LabelEncoder از scikit-learn برای تبدیل متغیرهای دسته‌ای به متغیرهای عددی استفاده می‌کند. ابتدا دو متغیر دسته‌ای به نام‌های "Country" و "Status" تعریف شده و سپس برای هر کدام از این متغیرها، ``LabelEncoder بر روی داده‌های مربوطه اعمال می‌شود.

سپس داده‌هایی که دارای مقدار ``NaN هستند با استفاده از ``data.dropna() حذف می‌شوند.

در نهایت، داده‌های تمیز شده با تبدیل متغیرهای دسته‌ای به متغیرهای عددی، به عنوان خروجی نمایش داده می‌شود.

```python
# Example: Assuming 'target_column' is your target variable
X = data.drop('Life expectancy ', axis=1)
y = data['Life expectancy ']
X,y
```

در اینجا دو متغیر X و y تعریف شده‌اند. متغیر X داده‌های ورودی را شامل می‌شود که شامل تمامی ستون‌ها به جز ستون "Life expectancy" است. متغیر y نیز متغیر وابسته یا هدف را نشان می‌دهد که در اینجا ستون "Life expectancy" است.

در اینجا دو بار دستورهای مشابه ارائه شده‌اند که متغیرهای X و y را نشان می‌دهند. این احتمال دارد که یک بار اضافی باشد. شما می‌توانید یکی از این دستورها را حذف کنید تا کد به درستی اجرا شود.

مشاهده خروجی:

```
(    Country  Year  Status  Adult Mortality  infant deaths  Alcohol  \
 0         0  2015       1            263.0             62     0.01
 1         0  2014       1            271.0             64     0.01
 2         0  2013       1            268.0             66     0.01
 3         0  2012       1            272.0             69     0.01
 4         0  2011       1            275.0             71     0.01
 ...     ...   ...     ...              ...            ...      ...
```

```
2933     192  2004      1          723.0           27     4.36
2934     192  2003      1          715.0           26     4.06
2935     192  2002      1           73.0           25     4.43
2936     192  2001      1          686.0           25     1.72
2937     192  2000      1          665.0           24     1.68

      percentage expenditure  Hepatitis B  Measles    BMI   ...  Polio  \
0                  71.279624         65.0     1154   19.1  ...    6.0
1                  73.523582         62.0      492   18.6  ...   58.0
2                  73.219243         64.0      430   18.1  ...   62.0
3                  78.184215         67.0     2787   17.6  ...   67.0
4                   7.097109         68.0     3013   17.2  ...   68.0
...                      ...          ...      ...    ...  ...    ...
2933                0.000000         68.0       31   27.1  ...   67.0
2934                0.000000          7.0      998   26.7  ...    7.0
2935                0.000000         73.0      304   26.3  ...   73.0
2936                0.000000         76.0      529   25.9  ...   76.0
2937                0.000000         79.0     1483   25.5  ...   78.0

      Total expenditure  Diphtheria    HIV/AIDS         GDP  Population  \
0                  8.16        65.0        0.1  584.259210  33736494.0
1                  8.18        62.0        0.1  612.696514    327582.0
2                  8.13        64.0        0.1  631.744976  31731688.0
3                  8.52        67.0        0.1  669.959000   3696958.0
4                  7.87        68.0        0.1   63.537231   2978599.0
...                 ...         ...        ...         ...         ...
2933               7.13        65.0       33.6  454.366654  12777511.0
2934               6.52        68.0       36.7  453.351155  12633897.0
2935               6.53        71.0       39.8   57.348340    125525.0
2936               6.16        75.0       42.1  548.587312  12366165.0
2937               7.10        78.0       43.5  547.358878  12222251.0

       thinness  1-19 years   thinness 5-9 years  \
0                      17.2                17.3
1                      17.5                17.5
2                      17.7                17.7
3                      17.9                18.0
4                      18.2                18.2
...                     ...                 ...
2933                    9.4                 9.4
2934                    9.8                 9.9
2935                    1.2                 1.3
2936                    1.6                 1.7
2937                   11.0                11.2

      Income composition of resources  Schooling
0                               0.479       10.1
1                               0.476       10.0
2                               0.470        9.9
3                               0.463        9.8
4                               0.454        9.5
...                               ...        ...
2933                            0.407        9.2
```

```
2934                    0.418       9.5
2935                    0.427      10.0
2936                    0.427       9.8
2937                    0.434       9.8

[1649 rows x 21 columns],
0      65.0
1      59.9
2      59.9
3      59.5
4      59.2
       ...
2933   44.3
2934   44.5
2935   44.8
2936   45.3
2937   46.0
Name: Life expectancy , Length: 1649, dtype: float64)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

در اینجا دو بار دستورهای مشابه برای تقسیم داده‌ها به دو قسمت آموزش و آزمون ارائه شده‌اند. این دستورات از ``train_test_split از scikit-learn برای تقسیم داده‌ها به دو قسمت استفاده می‌کنند. در اینجا، داده‌ها به ۸۰% برای آموزش و ۲۰% برای آزمون تقسیم می‌شوند و ``random_state برای تنظیم seed برای تولید اعداد تصادفی استفاده می‌شود.

همچنین، اندازه داده‌های آموزش و آزمون به همراه اندازه متغیرهای وابسته ((yنمایش داده شده است.

مشاهده خروجی:

```
((1319, 21), (330, 21), (1319,), (330,))
```

پایان